

The background is a vibrant yellow with a complex, organic pattern of overlapping circles and ovals in various shades of yellow and white. A large, white, stylized 'e' shape is superimposed over the background, partially enclosing the text.

Lotus

Enterprise Integrator

3.1
RELEASE

**Domino Connector
LotusScript Extensions Guide**

The background of the cover features a large, stylized, light gray graphic that resembles a large, open parenthesis or a stylized letter 'C'. Inside this shape, there are several smaller, overlapping, light gray oval shapes, some of which contain faint, abstract patterns. The overall design is clean and modern, with a focus on geometric shapes and a monochromatic color scheme.

Lotus

Enterprise Integrator

3.1

RELEASE

**Domino Connector
LotusScript Extensions Guide**

COPYRIGHT

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or part, without the prior written consent of Lotus Development Corporation.

© Copyright 2000 Lotus Development Corporation, an IBM subsidiary
55 Cambridge Parkway
Cambridge, MA 02142

All rights reserved. Printed in the United States.

Domino, Notes, NotesBench, Domino Connectors and Domino Enterprise Connection Services, Lotus Enterprise Integrator, LotusScript, Lotus NotesView, and Notes/FX are trademarks and Lotus, Lotus Notes, LotusScript, Notes Mail, NotesSQL, and NotesView, are registered trademarks of Lotus Development Corporation.

AIX, AS/400, DB2, DPROPR, OS/400, IBM, MVS, OS/2, OS390, S390, Presentation Manager, and SNA are registered trademarks, and DB2/2, RS/6000, OS/2 Warp, and PowerPC are trademarks of International Business Machines Corporation. Tivoli/Courier is a trademark of Tivoli Systems Inc., a wholly owned subsidiary of International Business Machines Corporation.

All other trademarks are the property of their respective owners.

Contents

| | | | |
|---|----------|--|-----------|
| 1 Introduction | 1 | 2 LCCONNECTION Class | 23 |
| Organization of This Manual | 1 | Overview | 23 |
| Overview of the LotusScript Extension for Lotus Domino Connectors | 2 | LCCONNECTION Properties | 24 |
| Connectivity Software Requirements | 3 | Common Connector Properties | 25 |
| Loading and Registering the LC LSX or LEI LSX | 4 | LCCONNECTION Class Methods Summary | 26 |
| Terms and Concepts | 5 | Connector Properties | 27 |
| LC LSX and LEI LSX Classes | 6 | New Method for LCCONNECTION | 28 |
| LCSession | 6 | Action Method for LCCONNECTION | 29 |
| LCCONNECTION | 7 | Call Method for LCCONNECTION | 31 |
| LCFieldlist | 7 | Catalog Method for LCCONNECTION | 32 |
| LCField | 7 | Connect Method for LCCONNECTION | 36 |
| LCStream | 7 | Copy Method for LCCONNECTION | 37 |
| LCNumeric | 8 | Create Method for LCCONNECTION | 38 |
| LCCurrency | 8 | Disconnect Method for LCCONNECTION | 41 |
| LCDatetime | 8 | Drop Method for LCCONNECTION | 42 |
| Working with the LSX | 8 | Execute Method for LCCONNECTION | 43 |
| Connection Pooling | 13 | Fetch Method for LCCONNECTION | 45 |
| Connection Pooling Overview | 13 | GetProperty Method for LCCONNECTION | 47 |
| Sequence of Events in Connection Pooling | 15 | GetProperty<Type> Methods | 48 |
| LSX Usage Notes | 17 | Insert Method for LCCONNECTION | 53 |
| LSX Data Types | 17 | ListProperty Method for LCCONNECTION | 55 |
| Arrays and Indexes | 18 | LookupProperty Method for LCCONNECTION | 58 |
| Working with Multiple Rows of Data | 19 | Remove Method for LCCONNECTION | 60 |
| Optional Parameters | 19 | Select Method for LCCONNECTION | 62 |
| Performance | 20 | SetProperty Method for LCCONNECTION | 64 |
| NOTES.INI Variables | 21 | SetProperty<Type> methods for LCCONNECTION | 65 |
| LEI.INI Variables | 22 | Update Method for LCCONNECTION | 69 |

| | | | |
|--|-----|--------------------------------------|-----|
| 3 LCCurrency Class | 73 | GetDatetime Method for LCField | 105 |
| Overview | 73 | GetFieldlist Method for LCField | 106 |
| Type CURRENCY Format and Values | 74 | GetFloat Method for LCField | 107 |
| LCCurrency Class Methods Summary | 74 | GetFormatDatetime Method for LCField | 108 |
| LCCurrency Properties | 75 | GetFormatNumber Method for LCField | 109 |
| New Method for LCCurrency | 75 | GetFormatStream Method for LCField | 110 |
| Add Method for LCCurrency | 75 | GetInt Method for LCField | 111 |
| Compare Method for LCCurrency | 76 | GetNumeric Method for LCField | 112 |
| Copy Method for LCCurrency | 77 | GetStream Method for LCField | 113 |
| Subtract Method for LCCurrency | 79 | IsNull Method for LCField | 114 |
| 4 LCDatetime Class | 81 | LookupVirtualCode Method for LCField | 115 |
| Overview | 81 | SetCurrency Method for LCField | 116 |
| Type DATETIME Format and Values | 82 | SetDatetime Method for LCField | 116 |
| LCDatetime Class Methods Summary | 83 | SetFieldlist Method for LCField | 117 |
| LCDatetime Properties Summary | 83 | SetFloat Method for LCField | 119 |
| New Method for LCDatetime | 84 | SetFormatDatetime Method for LCField | 119 |
| Adjust Method for LCDatetime | 85 | SetFormatNumber Method for LCField | 121 |
| Clear Method for LCDatetime | 86 | SetFormatStream Method for LCField | 122 |
| Compare Method for LCDatetime | 87 | SetInt Method for LCField | 123 |
| Copy Method for LCDatetime | 88 | SetNumeric Method for LCField | 123 |
| GetDiff Method for LCDatetime | 89 | SetStream Method for LCField | 124 |
| SetConstant Method for LCDatetime | 90 | SetVirtualCode Method for LCField | 125 |
| SetCurrent Method for LCDatetime | 91 | 6 LCFieldlist Class | 127 |
| 5 LCField Class | 93 | Overview | 127 |
| Overview | 93 | Accessing Field Data | 128 |
| LCField Class Methods Summary | 94 | Fieldlist Merging and Mapping | 129 |
| Processing Attachments Stored in RTF Fields | 98 | LCFieldlist Class Methods Summary | 130 |
| New Method for LCField | 99 | New Method for LCFieldlist | 131 |
| ClearVirtualCode Method for LCField | 99 | Append Method for LCFieldlist | 132 |
| Compare Method for LCField | 100 | Copy Method for LCFieldlist | 134 |
| Convert Method for LCField | 102 | CopyField Method for LCFieldlist | 135 |
| Copy Method for LCField | 103 | CopyRef Method for LCFieldlist | 136 |
| GetCurrency Method for LCField | 104 | GetField Method for LCFieldlist | 138 |

| | | | |
|---|-----|--|-----|
| GetName Method for LCFieldlist | 139 | LookupMetaConnector Method for LCSession | 187 |
| IncludeField Method for LCFieldlist | 141 | Sleep Method for LCSession | 191 |
| Insert Method for LCFieldlist | 142 | 9 LCStream Class | 193 |
| List Method for LCFieldlist | 144 | Overview | 193 |
| Lookup Method for LCFieldlist | 146 | LCStream Type TEXT Format and Values . . | 194 |
| Map Method for LCFieldlist | 148 | LCStream Type BINARY Format and Values | 195 |
| MapName Method for LCFieldlist | 150 | Stream Flags | 195 |
| Merge Method for LCFieldlist | 152 | Stream Format | 196 |
| MergeVirtual Method for LCFieldlist | 155 | New Method for LCStream | 197 |
| Remove Method for LCFieldlist | 156 | Clear Method for LCStream | 198 |
| Replace Method for LCFieldlist | 158 | Append Method for LCStream | 199 |
| SetName Method for LCFieldlist | 159 | Compare Method for LCStream | 200 |
| 7 LCNumeric Class | 161 | Convert Method for LCStream | 201 |
| Overview | 161 | Copy Method for LCStream | 202 |
| Type NUMERIC Format and Values . . . | 162 | Extract Method for LCStream | 203 |
| LCNumeric Class Properties Summary . | 162 | Merge Method for LCStream | 204 |
| LCNumeric Class Methods Summary . . | 163 | ResetFormat Method for LCStream | 205 |
| New Method for LCNumeric | 163 | SetFormat Method for LCStream | 206 |
| Add Method for LCNumeric | 164 | Trim Method for LCStream | 207 |
| Compare Method for LCNumeric | 165 | DatetimeListGetRange Method for LCStream | 208 |
| Copy Method for LCNumeric | 166 | DatetimeListGetValue Method for LCStream | 209 |
| Subtract Method for LCNumeric | 167 | DatetimeListInsertRange Method for LCStream | 210 |
| 8 LCSession Class | 169 | DatetimeListInsertValue Method for LCStream | 211 |
| Overview | 169 | DatetimeListRemoveRange Method for LCStream | 212 |
| LCSession Class Properties Summary | 170 | DatetimeListRemoveValue Method for LCStream | 213 |
| LCSession Class Methods Summary | 171 | NumberListGetRange Method for LCStream | 214 |
| New Method for LCSession | 172 | NumberListGetValue Method for LCStream | 215 |
| ClearStatus Method for LCSession | 172 | NumberListInsertRange Method for LCStream | 215 |
| GetStatus Method for LCSession | 173 | NumberListInsertValue Method for LCStream | 216 |
| GetStatusText Method for LCSession | 175 | | |
| ListConnector Method for LCSession | 176 | | |
| ListMetaConnector Method for LCSession . . | 180 | | |
| LookupConnector Method for LCSession . . | 184 | | |

| | | | |
|--|------------|--|------------|
| NumberListRemoveRange Method for LCStream | 217 | OLE DB Connector Properties | 268 |
| NumberListRemoveValue Method for LCStream | 218 | Sybase Connector Properties | 272 |
| TextListFetch Method for LCStream | 219 | Appendix D Character Sets | 276 |
| TextListInsert Method for LCStream | 220 | List of Supported Character Sets | 276 |
| TextListRemove Method for LCStream | 221 | Appendix E Additional Scripting Capability within LEI LSX | 285 |
| Appendix A Error Messages | 223 | Overview of Expanded Classes | 285 |
| Introduction | 223 | Expanded LCSession Functionality | 286 |
| Anatomy of an Error Message | 223 | Named Sessions, Named Connection Documents, and the LEI LSX | 287 |
| Example Error Message | 223 | AddProperty Method for LCSession | 287 |
| Error Message Components | 223 | GetCommand Method for LCSession | 289 |
| Format of Errors | 224 | GetProperty Method for LCSession | 289 |
| General Errors | 224 | GetProperty<Type> Method for LCSession | 290 |
| Connector Errors | 225 | ListProperty Method for LCSession | 291 |
| Parameterized Connector Errors | 232 | LookupProperty Method for LCSession | 292 |
| Field-Specific Parameterized Connector Errors | 235 | Log Method for LCSession | 293 |
| Appendix B Connector Property Tokens | 239 | LogEx Method for LCSession | 293 |
| Predefined Tokens | 239 | Usage | 294 |
| Connector Identification Property Tokens | 241 | LogText Method for LCSession | 295 |
| Appendix C Connector Properties | 243 | LogTextEx Method for LCSession | 296 |
| Connector Properties | 243 | Usage | 297 |
| Notes Connector Properties | 244 | RunActivity Method for LCSession | 298 |
| Notes Connector Property Combinations | 251 | SetProperty Method for LCSession | 299 |
| Processing Attachments Stored in RTF Fields | 251 | SetProperty<Type> Method for LCSession | 299 |
| Notes Virtual Fields | 252 | | |
| DB2 Connector Properties | 254 | | |
| EDA/SQL Connector Properties | 257 | | |
| File System Connector Properties | 258 | | |
| ODBC Connector Properties | 259 | | |
| Oracle 7 Connector Properties | 262 | | |
| Oracle 8 Connector Properties | 265 | | |

Chapter 1

Introduction

This chapter provides an introduction to the LotusScript Extension for Lotus Domino Connectors and gives information on the organization of this manual.

This chapter also gives a glossary of common terms and concepts used throughout the documentation, a general description of the individual classes of the LotusScript Extension for Lotus Domino Connectors, and describes how the classes might be used to implement a typical application. The documentation assumes a working knowledge of LotusScript, the Notes development environment, and the Notes classes. For more information on any of these subjects, please refer to the *Notes Programmer's Guide*.

Organization of This Manual

The table below describes the organization of this documentation and the information contained in each section.

| <i>Chapter</i> | <i>Description</i> |
|------------------------------------|---|
| Chapter 1 Introduction | This chapter provides an introduction to the LotusScript Extension for Lotus Domino Connectors. It also includes information about the organization of this manual. |
| Chapter 2 LCConnection Class | This chapter provides descriptions of the LCConnection Class methods and properties, and includes examples for using each of the methods. |
| Chapter 3 LCCurrency Class | This chapter provides descriptions of the LCCurrency Class methods and properties, and includes examples for using each of the methods. |
| Chapter 4 LCDatetime | This chapter provides descriptions of the LCDatetime Class methods and properties, and includes examples for using each of the methods. |
| Chapter 5 LCField Class | This chapter provides descriptions of the LCField Class methods and properties, and includes examples for using each of the methods. |

continued

| <i>Chapter</i> | <i>Description</i> |
|--|---|
| Chapter 6 LCFieldlist Class | This chapter provides descriptions of the LCFieldlist Class methods and properties, and includes examples for using each of the methods. |
| Chapter 7 LCNumeric Class | This chapter provides descriptions of the LCNumeric Class methods and properties, and includes examples for using each of the methods. |
| Chapter 8 LCSession Class | This chapter provides descriptions of the LCSession Class methods and properties, and includes examples for using each of the methods. |
| Chapter 9 LCStream Class | This chapter provides descriptions of the LCStream Class methods and properties, and includes examples for using each of the methods. |
| Appendix A Error Messages | This appendix provides a list of error messages that can occur during a script execution, and provides a description of the error message format. |
| Appendix B Property Tokens | This appendix provides a list of property tokens that are used in some of the LotusScript Extension for Lotus Connectors methods. |
| Appendix C Connector Properties | This appendix provides a list of the properties for each Lotus Connector. |
| Appendix D Character Sets | This appendix provides a list of character sets supported for use with the LotusScript Extension for Lotus Connectors. |
| Appendix E Additional Scripting Capability within LEI LSX | This appendix provides information about expanded LCSession functionality and additional class methods provided within LEI LSX. |

Overview of the LotusScript Extension for Lotus Domino Connectors

Lotus Domino Connectors provide native access to a wide variety of DBMS products, ODBC, the platform File system, Enterprise Resource Planning systems, and Transaction Processing systems. The LotusScript Extension for Lotus Domino Connectors (LC LSX for DECS and LEI LSX for LEI) extends these Connectors to LotusScript.

LotusScript provides an integral programming interface to Lotus Notes. The LotusScript Extension for Lotus Domino Connectors enhances the power of Notes by extending its scripting to data outside of Notes. The programming model is independent of the individual Connector. This eliminates the need

to learn each individual system, while at the same time allowing experienced users to access the individual features of a specific system.

For example, through Lotus Connectors, Notes and Web applications have the ability to retrieve and act upon data within agents, during document events, or at the click of a button.

This release of the LSX supports access to the following connectors:

- Notes
- DB2
- EDA/SQL
- File System
- ODBC
- Oracle 7
- Oracle 8
- OLE DB
- Sybase

In addition to these connectors, the Lotus Connector LSX will also support additional premium connectors, such as SAP, PeopleSoft, and so on. These specialized connectors may require additional methods; see the documentation that ships with each of these connectors for more information.

It is important to note that the LotusScript Extension for Lotus Domino Connectors may be used alone or in conjunction with the Domino Enterprise Connection Services (DECS) or Lotus Enterprise Integrator (LEI). Respectively, these technologies provide programmatic and declarative access to external data for application development.

Connectivity Software Requirements

Access to supported Lotus Connectors may require software to be installed on the Domino Server or the Notes client from which the Lotus Connector scripts are run. Refer to the *LEI Domino Connectivity and Installation Guide* for information about software that may be required in order to access any particular data source.

Note The *LEI Domino Connectivity and Installation Guide* is provided online in PDF format. The Connectivity Guide and the Installation Guide are supplied as separate .nsf files with the LEI product.

Loading and Registering the LC LSX or LEI LSX

The LC LSX is used by DECS. The LEI LSX is used by LEI. The LEI LSX contains all the current functionality of Domino LC LSX and more. As of LEI Version 3.1, LEI no longer uses the LC LSX; instead it uses the new LEI LSX. If you have LEI 3.1 and higher loaded, it automatically uses the (n)lsxlei.dll for LSX scripting; not the (n)lsxlc.dll. DECS continues to use the LC LSX.

You must load the LotusScript Extensions for Lotus Connectors using the following statement in your script:

For Domino/DECS users:

```
Uselsx "*lsxlc"
```

For LEI users:

```
Uselsx "*lsxlei"
```

Note As of Domino Release 5, Domino no longer registers the LC LSX when the Domino Server is installed. The LotusScript interpreter inside Domino (and Notes) detects the "*" at the start of the lsx name and knows how to perform necessary name construction to find the LSX – see sample syntax above.

For LEI Users Only

This manual describes the LSX for both DECS and LEI users. However, the LEI LSX contains unique functionality not available in the LC LSX. The following sections, as well as Appendix E, pertain to LEI users only.

Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these example show the following statement, which calls the LC LSX used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, LEI uses the LEI LSX. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

Named Sessions, Connection Documents, and the LEI LSX

The LEI LSX requires that you provide a name for every session and the name of an existing connection document. Example statement syntax is shown below:

```
Uselsx "*lsxlei"
```

```
DIM mySession As new LCSession ("MyLEISession")
```

```
Dim MyOracleConnection as New LCConnection("MyOracleConn")
```

In the above example, "MyOracleConn" is assumed to be an existing named connection document in the LEI Administrator. LC Session must now be named because this name is used as the default Log Document name. Prior to LEI Release 3.1, LEI used the LC LSX, which did not support a name argument in the DIM x as new LCSession statement. Existing LEI users should make note of this new behavior and provide a name for all LC Sessions.

Note One benefit of named sessions is that they enable the LEI LSX to support activity log documents.

Note The LEI Administrator uses the LEI LSX to perform various browsing calls, such as Select Metadata and Map Fields.

Terms and Concepts

Metadata

This is a generic term referring to a Connector's data definition. The data definition includes the names of data elements, their datatypes, and implies the order of the elements. For example, Notes uses 'forms' to describe both the names of data fields as well as the data type of each field; Sybase metadata describes a 'table.' Refer to Appendix C, "Connector Properties," for information specific to each Connector metadata.

Alternate Metadata

Available through some Lotus Connectors, this is an alternate source for the data definition. For example, DB2 metadata is in the form of a 'table', while its alternate metadata form is a "SQL view." Refer to Appendix C, "Connector Properties", for information on whether a specific Connector supports alternate metadata.

Result Set

The return from an information request through a connection produces a result set. Each connection can have a single active result set. The LCConnection methods Execute, Select, Call and Catalog each generate a result set, replacing any existing result set. The result set describes the collection of data or information from the connection, which matched the input criteria.

It does not return the actual data until fetched. If desired, these methods will build a fieldlist representing the metadata as part of generating the result set.

When using other connection methods which read or write data of a result set, the implied order of the metadata may be suspended by using the connection's `OrderBy` property to indicate that, regardless of the order of the data elements within the metadata, match the names in the metadata to the names in the external system.

Writeback Result Set

A Writeback result set is an optimized form of result set supported by some Connectors. A Writeback result set provides both sequential read and write operations on the data, and may be used for efficient Update and Remove operations by directly operating on the most recently fetched record in the result set, rather than having to locate the information in the external system a second time. Some Connectors may implement locking in the back end for Writeback result sets.

Token

A token is an integer used to identify a property of a connection. All connection properties have a token value. Common properties have predefined tokens represented by constants. Connection-specific properties do not have predefined tokens. (For a list of property tokens and names for each connection, see Appendix C.) Connection properties may be accessed by token or by name. The token method may be used when testing if a property is supported. The name may be used when it is known that a given property exists.

LC LSX and LEI LSX Classes

The Lotus Connectors provides external data and system access to the Domino LotusScript environment. The LSX classes consist of `LCConnection`, `LCFieldlist` and `LCField`, and four advanced datatypes, `LCStream`, `LCNumeric`, `LCCurrency`, and `LCDatetime`. In addition to these seven classes, there is also an `LCSession` class. Each of these classes and their primary usage is described below.

LCSession

The `LCSession` class provides error information useful in error handlers. It also provides for query and lookup of available Lotus Connectors.

Note The LCSession class has additional capabilities for LEI users within the LEI LSX. For example, using LCSession in LEI LSX enables you to create a new session. These additional capabilities are described in Appendix E, “Additional Scripting Capability within LEI LSX.”

LCConnection

The LCConnection class represents an instance of a Lotus Connector. This class provides query and data access to the external system. Multiple connections may be allocated to a single Connector.

LCFieldlist

The LCFieldlist class is the primary class for manipulating data through a connection. It binds a group of fields together with names and an implied order.

Fieldlists are used primarily for a number of connection operations; result sets and selection criteria, as well as reading and writing data.

When a result set is generated, and an empty fieldlist was initially passed in, the fieldlist is automatically populated by the Connector. For each data element of the result set, the fieldlist receives the element's name and a field object of the corresponding datatype. The result set may be controlled by manually building the fieldlist before the result set is constructed or by using the FieldNames property of the connection.

The Select and Call connection methods use an optional fieldlist of keys or parameters to restrict the result set. This fieldlist is manually constructed and passed to the select method. A key or parameter list is constructed by appending or inserting names and datatypes to the list. These methods create fields in the fieldlist and return these fields for further manipulation. These fields are then given values and, using field flags, may be given conditions such as greater-than, not-equal, etc.

LCField

LCField is the storage class that contains one or more data values. The datatype of a field is for all values contained within and may be any of the four advanced datatypes below, as well as long integer and double precision floating point, and, in some advanced usage, fieldlist or connection.

LCStream

LCStream is a general-purpose text and binary datatype. The contents of a stream are marked with a format that details the character set of the text or any special attributes of the binary data.

LCNumeric

The LCNumeric class is a container for very high precision numbers.

LCCurrency

The LCCurrency class is a fixed point decimal datatype with 4 decimal places and 19 digits of precision. (This is mathematically equivalent to the LotusScript datatype and is provided to support connections with a dedicated currency.)

LCDatetime

The LCDatetime class is a date and time datatype which is accurate to the hundredth of a second and which is aware of time zones and daylight savings time.

Working with the LSX

A typical use of the Lotus Connectors is to gather, create, or modify data in an external system. For example, a Notes application has a number of data fields on a form. Once the user input is complete, a button activates script to take the form data and establish a connection to Oracle, locating corresponding information from one or more tables and updating the Notes form. The script is responsible for:

1. Accessing the active Notes form
2. Gathering the input data
3. Creating a connection to the external system
4. Selecting the data based on the input values
5. Loading the data from the external system
6. Storing the results in the active Notes form

Here is a simple script to accomplish the task. The assumption is that the Notes form has a single text input field called "Customer". The script will use the value of the customer field to locate the corresponding customer information in DB2 and return an address and phone number for the customer storing the return values in the form using fields called "Address", "City", "State", and "Phone".

Note No attempt has been made to prescribe a code style. The practice of grouping object declarations together at the beginning of a script versus locating declarations close to the code is a preference and does not affect the execution. In this example, declarations and code are grouped to facilitate explaining the process.

The first step in writing the script is to load the LotusScript Extensions for Lotus Connectors. The `Uselsx` statement accomplishes this step. Additional options may be used to check variables, simplify string comparison, and so on. (In the example that follows, “*” refers to the operating system prefix for *lsxlc or *lsxlei.)

```
Option Public
Option Explicit
```

```
Uselsx "*lsxlc" //For DECS
```

or

```
Uselsx "*lsxlei" //For LEI
```

Note For LEI you must also `Dim LCSession` as a named session (this becomes the default Log Document name) and `LCConnections` must be dimensioned as existing named connection documents that have been created in the LEI Administrator. See the overview section of each of the method chapters for more information and an example.

The remainder of the script is located in the ‘Click’ event of the form’s button. Errors should be displayed to the user. A simple error handler is written at the bottom of this example. The LSX Session class has a `Status` property that may be used to determine if the error handler was triggered by an LSX error or a LotusScript error. In all cases where the LSX reports an error, the LotusScript “Error\$” will contain error information. However, when first creating LSX objects, the LSX has additional error information not available through the LotusScript error statements. Creating and initializing the Session status provides this additional information for the error handler.

The creation of the session object is not necessary for normal error handling.

```
Sub Click (Source as Button)
    On Error Goto Handler
    Dim session as New LCSession
    session.ClearStatus
```

The input values are in the current active document. This information is accessible via the `NotesUIDocument`, which may be located through the `NotesUIWorkspace` from its “CurrentDocument” property.

```
Dim wksp As New NotesUIWorkspace
Dim uidoc As NotesUIDocument
Set uidoc = wksp.CurrentDocument
```

The next step establishes a connection to the Lotus Connector for DB2. After the connection has been created, all of its properties are accessible to customize the connection to the target system. Common properties include

Database and/or Server, UserID, and Password. Properties are not case sensitive. (For a complete list of properties for each Lotus Connector, see Appendix B.) The following code connects to the DB2 system called Rainbow as *jdoe* with the password *gold*.

```
Dim src As New LCConnection ("db2")
src.Database = "Rainbow"
src.UserID = "jdoe"
src.Password = "gold"
src.Connect
```

There are four connection methods for querying through a connection: Catalog, Execute, Select and Call. The catalog operation is used to return metadata information within the external system, for example, the tables of a DB2 database or the columns of a specific Sybase table. For a complete list of Catalog options, see the Catalog method. The remaining methods, Execute, Select and Call, create result sets of data from the connection. The methods differ significantly in their interface. The execute statement uses a connector-specific command statement to determine the contents of the result set. This interface is helpful when the external system's command structure is familiar and when cross-connector portability is not an issue. The Select method uses a combination of key names, values, and condition flags to indicate the desired contents of the result set. This interface works across connectors and does not require knowledge of the connector's command language. The Call method is similar to Select, but is used for calling back-end procedures or functions. Instead of keys, parameters are provided.

For our example, the important data are stored in the "Customer" DB2 table, as indicated by the Metadata property of the connection. The only record of interest is the customer named by the input value from the Notes form.

Customer selection is accomplished by creating a key list. The default key flag, LCFIELDF_KEY, indicates match exactly. If an inequality match such as 'greater-than' or 'like' is needed, then the field's flags property would be Ored with the corresponding constant. (In all cases, key fields must have the LCFIELDF_KEY constant in addition to any optional conditional flag constants.)

```
Dim keys As New LCFieldList
Dim field As LCField

src.Metadata = "Customer"
Set field = keys.Append ("Name", LCTYPE_TEXT)
field.Flags = LCFIELDF_KEY
field.Text = uidoc.FieldGetText ("Customer")
```

The Select connection method creates a result set of all records from the external system which match the keylist. If the LotusScript keyword “Nothing” is substituted for the key list, then all records of the specified metadata would be selected. In this case, all records from the “Customer” DB2 table would be selected. This example is interested in just the customer record matching the input value from the Notes form. The key list is created to make this restriction.

The fieldlist receiving the result set is currently empty. The selection will populate the fieldlist with the fields from the DB2 table. If all of the fields of the metadata are not needed, the result set may be restricted to just the fields of interest either by creating the fieldlist prior to the selection or by setting the ‘FieldNames’ property of the connection.

```
src.FieldNames = "Address, City, State, OfficePhone"
```

The selection returns one of three values: the number of records selected; zero (0) if no matching records were found; or LCCOUNT_OVERVIEW, when records were found but the connection does not know the total. Since a return of zero is the only case where data was not found, it is the test case for error handling or branching.

```
Dim fields As New LCFieldList
If (src.Select (keys, 1, fields) = 0) Then End
```

A result set has been created and there is a match. The result set has not retrieved the data. The Fetch connection method reads the data from the external system into the fieldlist. The individual data values may be accessed from a fieldlist using the expanded class properties. For each field in a fieldlist, there is a property with the corresponding name. This property is an array of values using the closest available LotusScript datatype to match the LSX LC datatypes.

```
If (src.Fetch (fields) > 0) Then
    Call uidoc.FieldSetText ("Address",
fields.Address(0))
    Call uidoc.FieldSetText ("City", fields.City(0))
    Call uidoc.FieldSetText ("State", fields.State(0))
    Call uidoc.FieldSetText ("Phone",
fields.OfficePhone(0))
End If
```

Note When writing scripts that act on more than one record, it is more efficient to locate the field from within the fieldlist, outside the loop, and then use the field for data access. Using the expanded class properties locates the field each time it is used and allocates an array of values, not just a single value. Here is a sample of this code. The %REM has been used to indicate that this code is not part of the actual example.

```

%REM
    Dim address as LCField
    Dim city as LCField
    Dim state as LCField
    Dim phone as LCField

    Set address = fieldlist.Lookup ("Address")
    Set city = fieldlist.Lookup ("City")
    Set state = fieldlist.Lookup ("State")
    Set phone = fieldlist.Lookup ("OfficePhone")

    while (src.Fetch (fields) > 0) Then
        Call uidoc.FieldAppendText ("Address",
address.Text(0))
        Call uidoc.FieldAppendText ("City", city.Text(0))
        Call uidoc.FieldAppendText ("State", state.Text(0))
        Call uidoc.FieldAppendText ("Phone", phone.Text(0))

    Wend

%END REM

```

The data has been retrieved from the external system and placed in the Notes form. This completes this example. The final step is to refresh the Notes document to display the new data to the user.

```

    uidoc.Refresh
End

```

An error handler was designated as the first line of this example. Testing for an LSX error first provides additional information in the case of an object creation error. Without the session object and subsequent test in the error handler, failure while creating a connection to DB2 generates the LotusScript message, "Error creating product object." However, for the same error condition, the LSX reports "Error: Cannot load LSX library 'db2'."

```

Handler:
    If (Session.Status <> LCSUCCESS) Then
        MessageBox Session.GetStatusText, 0, "The following
Lotus Connector error has occurred"
    Else
        MessageBox Error$, 0, "The following LotusScript
error has occurred"
    End If
End
End Sub

```

This example is very simple. It is meant only to provide an understanding of the Lotus Connectors, the classes, and the relationship between the connection, metadata result set, fieldlist, fields and data.

Connection Pooling

Connection pooling is part of the common LC API. The LSX is currently the only technology which provides an end-user interface to the pooling code. The functionality is switched off by default, this minimizes any potential adverse behavior connection pooling might introduce to existing code.

Note The LSX operates in two different modes: standard and LEI. Connection pooling is available only when the LSX is loaded in standard mode. (LEI mode is indicated by the presence of an LEI installation and the existence of the LOTUS_CONNECTOR_INIT environment variable.)

Connection Pooling Overview

The LSX provides a simple programmatic interface to connection pooling. The LCSession class exposes a property called “ConnectionPooling.” Setting this property to “True” enables connection pooling.

A typical operation consists of three major parts (see below, “Sequence of Events for Connection Pooling,” for more details):

1. Connecting to the data source
2. Identifying information of interest, retrieving and or updating the information
3. Disconnecting from the data source

The middle step is the processing step. When you work with large amounts of information or when you are performing complex operations, processing becomes the most significant time consideration. Many applications have a very short processing step. In these applications, a much greater percentage of total time is associated with connecting to and disconnecting from the external data source. The connect/disconnect time is amplified when it must occur for each processing request.

The LSX for Lotus Domino Connectors provides a connection pooling property that makes it possible to retain discarded connections for later use. The pooling functionality is controlled by the LCSession property, ConnectionPooling, which is a Boolean with a default value of FALSE. When this property is set to TRUE, subsequent requests for new connections are processed through the connection pool.

With connection pooling, creating a new connection is serviced first by checking the pool for an existing compatible connection and then if one is not available, creating a new connection. A compatible connection is determined by the external system and all of the system’s required connectivity properties. This prevents a connection originally established for one user, later being used by another. As an example, a connection to DB2 which was originally created using John’s user name and password, and then release to

the connection pool would not be issued if a new connection was requested using Jane's user name and password.

A connection is removed from the pool when the "Connect" method occurs. A connection is returned to the pool when the "Disconnect" method occurs. If no explicit disconnect occurs, an automatic disconnect is performed when the object is deleted. Note: In LotusScript, if no explicit "delete" occurs on an object, it is automatically deleted when it falls out of the scope of the function, subroutine, or script.

Keep in mind that a connection that is returned to the pool does not disconnect from the external system. Code that takes advantage of connection pooling must anticipate this behavior. The important issues to remember are related to what may happen automatically during a normal disconnect from an external system. For example, disconnecting from an RDBMS may trigger a commit of records inserted, updated, or deleted, since the connection was first established. Likewise, there may be rollback or other database operations that take place automatically as a result of disconnecting. When connection pooling is enabled, these events do not take place because the connection is not actually dropped. Therefore, if you expect and want these types of operations to take place, the processing portion of the script must explicitly perform them.

The life span of a pooled connection is dependent on the LSX. Within Notes and Domino, an LSX is loaded when the execution of the `Uselsx` statement occurs in a script. The LSX is not unloaded until the Notes or Domino process terminates. Once a connection is pooled, it remains available until the associated process terminates.

The connection pool defaults to a maximum of 20 pooled connections for a given external system. When the maximum number of connections have been created and are already in use, a request for an additional connection will be granted but the connection will not be pooled. The default may be overridden using INI settings: for LEI use the `LEI.ini` file, for DECS use the `Notes.ini` file.

The syntax for configuring the connection pools is a list of comma-delimited Lotus Domino Connector names, the pool size, and an optional data source maximum. The default pool size for all connectors is 20. The optional data source maximum value indicates the limit of allowed connections to a single database, obviously this value cannot be more than the total pool size for a given connector. For example, a DB2 pool size of 10 and a data source maximum of 5 indicates the pool will hold no more than 10 connections to DB2 and of the 10, no more than 5 will be to any one database. If you do not specify a data source maximum, it is the same as the pool size.

Here is an example of an INI entry:

```
ConnectionPool=oracle,10,db2,20,5,psft7,4
```

This example has pooling information for three connectors: Oracle, DB2, and PeopleSoft 7.0. Respectively, they support connection pools of 10, 20, and 4 connections each. In addition, DB2 indicates that no more than 5 connections in the pool can be to a single database.

Note Connection pooling works for LotusScript agents because the LSX is loaded and maintained by the process space (that is, whatever process loaded the LSX: the server, the agent manager, http, or some other process). Also, if two agents use the RunOnServer method and they are run at separate times, they both execute in the same process space and pooling works.

Sequence of Events in Connection Pooling

Programs which use the LC API follow a typical progression:

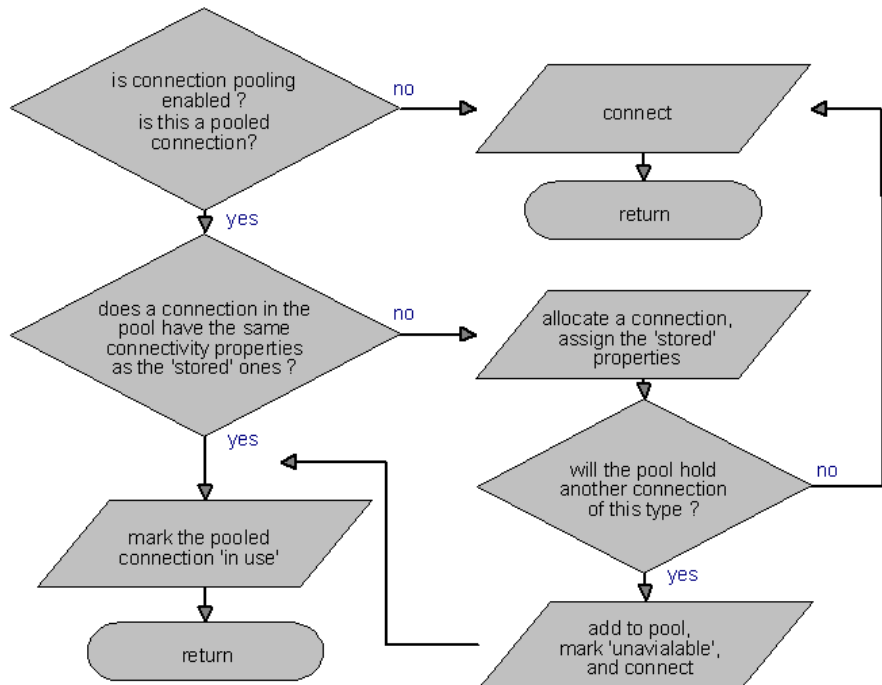
1. Allocate a connection
2. Set connectivity properties
3. Connect
4. Select data
5. Fetch data
6. Manipulate data
7. Store data
8. Disconnect
9. Free connection

Connection pooling primarily affects the first three steps during which the connection is established. While the presentation material contains material pertaining to each step, the diagram below details the connection pooling process.

One goal of connection pooling is to minimize any adverse impact on performance. For this reason, all code which utilizes a connection handle, must determine quickly if pooling is enabled and if a given connection handle refers to a traditional connection or a pooled connection.

The connection pool is subdivided based on connectors. There is a portion of the overall pool for each connector type used. The pool is configurable on a per connector basis. The default pool maintains 20 connections for any given connector. Connections are not established until requested by the users program. Additionally, new connections are not created unless no applicable connection is available in the pool. The pool may be configured

to limit the total number of connections for an individual connector as well as limit the number of identical connections. A connection's individuality is based on its connectivity parameters or credentials. Connections are considered to be identical when they are for the same connector and all of the connectivity properties match. The matching criteria is also used to determine if a connection in the pool satisfies a user request for a connection.



When a connection is first allocated, the connection pool is checked to verify that pooling is enabled and the pool has as least one connection of the requested type: DB2, Oracle, SAP, and so on.

- The pooling code creates a facsimile of a connection and returns its handle in place of a real connection handle. The purpose of this facsimile is to emulate a connection while the user assigns the connectivity properties. The properties must be kept separate from a real connection until the request to establish a connection to the external system.
- When connect request occurs, the properties held in the facsimile are used to locate a matching connection in the pool. Assuming one is found, it is returned. Otherwise, a new connection is created returned.

- If the pool is full, the newly created connection is not pooled. The LC API can quickly differentiate between pooled and non-pooled connection permitting them to be mixed freely. The user is unaware that a request for a pooled connection was actually served by an un-pooled connection. The only evidence would be the absence of any performance gains associated with pooling.

LSX Usage Notes

Use of the Lotus Connectors LotusScript Extensions is limited to the Notes environment and Notes applications. For example, LSX scripts cannot be called from within Lotus 1-2-3 or Lotus Approach, because the Connectors only have context within the Domino Server and Notes client.

LSX Data Types

Standard Data Types

Standard data types come in three classes: Number (Int, Float, Currency, and Numeric), Datetime (Datetime), and Stream (Text and Binary). All types except Int and Float are represented by Lotus Connector classes described in this manual (Int is represented by a LotusScript Long, and Float by a LotusScript Double).

- **Int:** A four-byte signed integer. Integers are very efficient, but limited in precision and range. An int is any whole number between -2147483647 and 2147483647 (9 digits of precision).
- **Float:** An eight-byte (double-precision) floating point number. Floats have less precision than currencies or numerics, but have a far greater range of values and are more efficient. A float has 15 digits of precision, and can represent any practical value (with an exponential range between 10^{308} and 10^{-308}).
- **Currency:** An eight-byte integer with four decimal places. Currency provides greater precision than either int or float values. While it is less precise than a numeric, it is far more efficient, making it the preferable choice when higher precision is required. A currency has 19 digits of precision, and a maximum value of 922,337,203,685,477.5807.
- **Numeric:** A numeric is larger and less efficient than any other number datatype, but has far greater precision. A numeric is assigned a precision (number of digits) up to 88 and a scale (number of decimal places) between -127 and 127, allowing it to express a very large number of values with very high precision.

- **Datetime:** A datetime indicates a specific date and time. Its range is any date between the years 1 and 32767, with a precision of .01 seconds. In addition, datetimes are specific to time zone and DST settings, making them usable in international settings.
- **Text:** Text is a stream of characters up to 4 Gb in length. All text has a character set, and conversion between character sets is supported. When transferring between systems, character set conversion is automatically performed as needed.
- **Binary:** Binary is a stream of bytes up to 4 Gb in length. All binary data has a format which is either BLOB (Binary Large Object) for unformatted data, or a structured binary format such as composite, text list, number list, and datetime list. The three list formats allow for a single data value to itself contain multiple values.

Advanced Data Types

Advanced data types comprise two areas: Fieldlist and connection.

- **Fieldlist:** A fieldlist itself is a collection of named fields with one or more values each, similar to a SQL table. A fieldlist as a data type allows for a single data value to itself contain a fieldlist, supporting hierarchical or nested data. The fields in a fieldlist data value may in turn contain additional nested fieldlists. Fieldlists are only usable by those back-end that support such data structures.
- **Connection:** A connection supports interaction with a back end system through a Lotus Connector. Connections as a data type are only valid when working with metaconnectors, since metaconnectors have properties of type Connection.

Arrays and Indexes

The default for array indexes in LotusScript is 0 based. For example, elements of an array are specified as Elem(0), Elem(1), Elem(2), etc. There is an option in LotusScript, "Option Base n," that makes the default lower bound of an unspecified array something other than 0. LSXs do not support this syntax. It is recommended, for consistency, that within scripts which use an LSX, such as the Lotus Connectors, that this option not be used.

Many Lotus Connector class methods operate on multi-value data. Class methods include the List methods of LCStream, the Get and Set methods of LCField and LCFieldlist, as well as the Fetch, Insert, Update, and Remove methods of LCConnection. These operations are counted from 1. For example, fields of a fieldlist are specified as MyFieldList.GetField (1), MyFieldList.GetField (2), MyFieldList.GetField (3), ... etc. Unlike dimensioned arrays which default to zero based, and LSX generated arrays which are always zero based, these class methods require an index counted from 1.

It is possible to assign a LotusScript array to a multi-value object such as a LCField. It is important that the array not contain more elements than the LCField object will store, as this will cause an overflow error. For example, 5 long integers may be assigned to an LCField as follows:

```
Dim numbers(4) as Long
    ' numbers(0, numbers(1), ... numbers(4)
Dim field as new LCField (LCTYPE_INT, 5)

...
field.Value = numbers
```

Working with Multiple Rows of Data

The Fetch, Insert, Update, Remove methods of a connection use a *Record-Count* parameter and may operate on more than a single row at a time. However, the LCFieldlist parameter must have been created to hold more than one record for this to work. You may operate on fewer records than the Fieldlist was created with, but not more.

A simple example of different configurations for *RecordIndex* and *Record-Count* might be as follows:

```
Dim Records as New LCFieldlist (3, LCFIELDF_TRUNC_DATA)
...
Call connection.Fetch (Records, 1, 3) ' fetch three records
Call connection.Insert (Records, 3, 1)
    ' insert them in reverse order
Call connection.Insert (Records, 2, 1)
Call connection.Insert (Records, 1, 1)
```

Note Both parameters for New LCFieldlist are optional and default to 1, and LCFIELDF_TRUNC_PREC, respectively. Likewise, the *RecordIndex* and *RecordCount* parameters are optional for Fetch, Insert, Update and Remove, and default to 1 and 1. Lastly, the *RecordIndex* and *RecordCount*, when added to each other (minus 1) must not exceed the size of the created Fieldlist.

Optional Parameters

Some methods have optional parameters. Note the LotusScript keyword “Nothing” is used for output parameters that the user is not interested in receiving. If one or more optional parameters are at the end of a call, they may simply be ignored as in the case of a datetime. For example, a datetime with only the date portion provided is created using the following statement.

```
Dim Calendar as New LcDatetime (1998, 8, 7)
```

When optional parameters that occur in the middle are omitted, the commas must remain.

```
Call session.GetStatus (ErrorText, , DBError)
```

Performance

Certain techniques can be used to increase LSX code performance. These techniques are described below.

Connector Caching

Most connectors are optimized for repeated calls using the same parameters, but different data values. This generally applies to Fetch, Insert, Update, Remove, and Select operations. When these methods are called with the same fieldlist more than once, they are able to avoid certain steps such as field mapping and type checking. This optimized functionality is available as follows:

Select: METADATA property and KeyFieldlist are the same.

Fetch: Result set and DestFieldlist are the same.

Insert: METADATA property and SrcFieldlist are the same.

Update and Remove: WRITEBACK and METADATA properties and SrcFieldlist are the same.

Calls to one type of method generally do not affect others, so the following pseudo-code obtains the optimal use of this caching:

```
Connect to Connection CONN  
  
Set Metadata  
  
Generate result set, producing Fieldlist FL  
  
Loop until done:  
  Fetch into FL  
  Insert from FL  
  
Exit
```

If this were changed to fetch the data and then inserted into one of two tables depending on some condition, then every time the table changed, new field mapping and type checking would be performed. In such a case, establishing two connections for Insert would be optimal.

Tight Loops

Another way to optimize LSX code is to produce tight loops, moving as much code out of the loop as possible. Anything inside such a loop must be done for every iteration of the loop, and can therefore become costly with large data sets. For example, inside a loop which transfers 10,000 records, you may need to access a field in a fieldlist. Instead of using the property as `Fieldlist.FieldName`, which will locate the field by name every iteration, you could instead use `Field = Fieldlist.FieldName` outside the loop and then simply refer to `Field` inside the loop. This will remove 10,000 property accesses for the fieldlist.

NOTES.INI Variables

You can use the `notes.ini` file to adjust aspects of Domino and DECS. The following entries may be added to the `notes.ini` file to control aspects of the LSX.

DECSTranslation

This controls text translation, allowing the user to increase performance in exchange for certain assumptions about the data being accessed. Note that none of these settings affects translation between unicode and other character sets, since it is always required. There are three valid numeric settings:

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| 0 | Do not perform translation between character sets (except unicode). This setting is valid when all data being accessed is compatible with the Notes LMBCS character set – primarily ASCII printable characters. |
| 1 | No not perform translation between non-LMBCS (and non-unicode) character sets. This setting is valid when all data being accessed, except for Notes LMBCS data, is in compatible character sets. |
| 2 | Always translate between any character sets. This is the default. |

DECSCenturyBoundary

This controls how to interpret the year in a text string being converted to a datetime, when the year contains only two digits. Values greater than or equal to the century boundary are considered to be in 1900s, values less than are in the 2000s. There are three ways to set this:

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| 0 | Since all values are greater than or equal to zero, always use 1900. |
| 1-100 | If the two-digit year is greater than or equal to this value, use 1900; otherwise use 2000. |
| 101 | Since all values are less than 101, always use 2000. |

The default setting is 50, which is the same as Notes. This means that any two-digit year from 0-49 is in 2000, and 50-99 is in 1900.

DECSNativeText

This allows the local machine's native character set to be overridden. There are various situations when the native character set is used within the LSX, and some back-end systems always consider client data to be in the native character set. Setting this value to a valid text format string replaces the character set obtained from the operating system by the LSX with the indicated character set. Appendix D lists supported character sets. Use the text that remains after removing the "LCSTREAMFMT_" prefix. For example, Code Page 932, represented by the constant LCSTREAMFMT_IBMCP932, would be set as follows:

```
DECSNativeText=IBMCP932
```

LEI.INI Variables

You can use the lei.ini file to adjust aspects of LEI. For example, you can force the local machine's native character set to be overridden by adding the following line to the lei.ini file:

```
NativeText=IBMCP932
```

The LCSTREAMFMT_IBMCP932 represents IBM character set code page 932. For more information on character sets within LEI, see Appendix C, "Character Sets" in the *Lotus Enterprise Integrator User Guide*.

For more information about lei.ini, see the LEI documentation set supplied with that product on the distribution CD-ROM.

Chapter 2

LCConnection Class

This chapter provides information about the Lotus Connectors LCConnection class methods and properties

Overview

The LCConnection class represents an instance of an individual Lotus connector. One connection object should exist for each individual data connection through a connector. This is true when the connections are to the same Lotus connector as well as to different connectors. For example, transferring data from one DB2 table to another table in Oracle is best accomplished by creating two LCConnection objects, one for DB2 and one for Oracle, and fetching data from one and inserting it into the other.

Connection methods manage individual Lotus connector connections within a session. The Connection class enforces connector state and requirements, and must always be used when interacting with Connectors. One connection object should exist for each individual data connection through a connector used in a session.

Error information for a connection is available through use of the LCSession class.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session (this becomes the default Log Document name)
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "*lsxlei"
```

Sub Initialize

```
Dim MyLEISession As New LCSession ("MainSession")
```

```
Dim MyNotesConnection as New LCConnection  
("MyNotesConn")
```

```
Dim MyOracleConnection as New LCConnection  
("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

LCConnection Properties

All Connections have a set of properties with values that can be assigned and retrieved. All properties are represented by both an integer value (called a property token) and a name. Some property tokens are valid for all connectors, while others are connector-specific. Refer to Appendix C for a list of properties for each Lotus connector.

All connector properties are of the closest LotusScript data types when you access connection properties using this form:

```
Value = LCConnection.<PropertyName>
```

or

```
LCConnection.<Property.Name> = value
```

When referencing the connector’s property by name, the property will be represented using the closest matching LotusScript data type; not the Lotus connector data type. For example, if the property were textual, the LC data type would be an LCStream, however, the syntax above would use the closest matching LotusScript data type which would be a “String”.

Likewise, an LCDatetime corresponds to a LotusScript “Variant” with date and/or time information.

Note When you use the LCConnection.GetPropertyXXXXXX() syntax, the data type will be an LC datatype.

Common Connector Properties

The following is a list of predefined properties common to all connectors. Use these predefined properties with the `LCCConnection.GetProperty` and `LCCConnection.SetProperty` methods. See “Miscellaneous” under the “LCCConnection Class Methods Summary” section below as well as the entry for each method.

| <i>Property</i> | <i>Name</i> | <i>Description</i> |
|-------------------------|------------------|---|
| LCTOKEN_NAME | “Name” | Connector name. |
| LCTOKEN_CONNECTOR_CODE | “ConnectorCode” | Connector Virtual Code. |
| LCTOKEN_CONNECTION_CODE | “ConnectionCode” | Connection Virtual Code. |
| LCTOKEN_EVENT_ERROR | “EventError” | Lotus connector status code to treat as a non-error event. See below. |
| LCTOKEN_TEXT_FORMAT | “ TextFormat” | Stream format constant for text data passed between the connector and the external system. This property is generally read-only. |
| LCTOKEN_CHARACTER_SET | “ CharacterSet” | Lotus connector character set indicator. See below. |
| LCTOKEN_IGNORE_ERROR | “IgnoreError” | Lotus connector status code to ignore. See below. |
| LCTOKEN_LCX_VERSION | “LCXVersion” | LCX version (from LCXIDENTIFY structure). |
| LCTOKEN_CONNECTOR_NAME | “ConnectorName” | Convenience property assigned to the first sub-connector for a metaconnector LCX. Setting this property to a connector name is equivalent to creating a connector of this type and setting the connector itself to the subconnector property. |
| LCTOKEN_IS_CONNECTED | “IsConnected” | TRUE following a successful Connect call, until Disconnect is called. FALSE otherwise. |

Properties of a connection may also be accessed by name, for example, the following line of script sets the value of Database property for a DB2 connector to "HR."

```
Connection.Database = "HR"
```

Note Connector-specific properties are listed in Appendix C, "Connector Properties," of this manual.

LCConnection Class Methods Summary

Connection

The following methods control connection to a data provider. A connection is required before gaining access to most connector metadata and all connector data. Multiple connections may be established to a single connector with multiple Connection objects.

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| LCConnection.Connect | Establish a connection to a data provider. |
| LCConnection.Disconnect | Disconnect from a data provider. |

Create Result Set

Each Connection can have a single active result set. A result set is the data produced by an action, for example, the execution of a Select statement against SQL tables. All of these methods produce a result set, replacing any existing result set. The result set can be produced from a connector-specific language statement, from Connector-independent properties and keys or parameters, or from connector metadata. connector metadata is either an SQL table or view, a Notes form, etc.

Once a result set is produced, the data in that result set can be fetched. Under specific circumstances, efficient writeback updates and removes directly back into the result set are also supported.

Note The Execute and Call methods invoke an operation from the external system.

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| LCConnection.Call | Perform a Connector-independent procedure call with parameters. |
| LCConnection.Catalog | Produce a result set containing a metadata catalog. |
| LCConnection.Execute | Execute a statement against the data provider. The statement is provided in the data provider's language. |
| LCConnection.Select | Perform a Connector-independent selection controlled by various connector properties. Conditional key inequalities, timestamp, and other control is supported. |

Data Manipulation

These methods allow access to and manipulation of connector data.

| <i>Value</i> | <i>Description</i> |
|---------------------|--|
| LCConnection.Fetch | Retrieve records from the current result set. |
| LCConnection.Insert | Insert new records into the data provider. |
| LCConnection.Update | Update records in the data provider. Key values and update values are provided. No keys are required for writeback operations. |
| LCConnection.Remove | Delete records from the data provider. Key values are provided. No keys are required for writeback operations. |

Metadata Manipulation

These methods allow access to and manipulation of connector metadata.

| <i>Value</i> | <i>Description</i> |
|---------------------|-----------------------------------|
| LCConnection.Create | Create a new metadata object. |
| LCConnection.Drop | Drop an existing metadata object. |

Miscellaneous

| <i>Value</i> | <i>Description</i> |
|--|---|
| LCConnection.Action | Perform one of a set of predefined actions. |
| LCConnection.GetProperty | Fetch a property value for a connection. |
| LCConnection.GetProperty <datatype> | Fetch a property as a particular data type. |
| LCConnection.ListProperty | List supported properties and values. |
| LCConnection.LookupProperty | Verify the support of a property. |
| LCConnection.SetProperty | Set a property value for a connection. |
| LCConnection.SetProperty <datatype> | Set a property as a particular data type. |

Connector Properties

Refer to Appendix C, “Connector Properties,” for more information about the properties for each Lotus connector.

New Method for LCConnection

This is the constructor for objects of class LCConnection.

Defined In

LCConnection

Syntax

Dim *connectionName* as New LCConnection(*libraryName*) // DECS only

or

Dim *connectionName* as New LCConnection(*ConnectionDocumentName*) // LEI only

Parameters

| <i>Parameter</i> | <i>Description</i> |
|-------------------------------|---|
| <i>libraryName</i> | The name of a valid, installed connector, such as db2 or oracle, and the db2.lcx or oracle.lcx exists in the Notes/Domino directory. Note Use lowercase letters for <i>libraryName</i> , as some file systems are case-sensitive. See the LCSession.ListConnectors method, which is used to determine the installed Connectors. |
| <i>ConnectionDocumentName</i> | The name of a valid Connection Document that exists in the LEI Administrator. |

Usage Notes

- A connection library is located by name. When a scripted agent is called, it calls a defined activity. The defined activity must have a named connection document associated with it. As a result, when the LCConnection object is instantiated with the following call, the <libraryname> takes on a new meaning in this context. The library name becomes the name of the connection that will be used by the script.

```
Dim <connectionname> as New LCConnection ( <libraryname> )
```

Using the LEI LSX, the new session is named when it is defined. All activities/sessions must be named. However, using the LC LSX, you can create an activity or session without naming it. This creates a nameless activity.

If a script is run without an activity association, as is possible using the LC LSX, then <libraryname> is the type of connector – such as db2, oracle, odbc2, and so on.

- The constructor allocates a code for this connection, unique among all connections, which can be used as a virtual code for a field (see `LCField..SetVirtualCode`). This value can be retrieved as the connector property `LCTOKEN_CONNECTION_CODE`.

Example

```
Dim MyNewConnectionName as New LCConnection("oracle")
    // DECS only
```

or

```
Dim MyNewConnection as New LCConnection("OracleConnDoc")
    // LEI only
```

Action Method for LCConnection

This method performs an action as defined by the `actionType` parameter.

Defined In

LCConnection

Syntax

Call *lcConnection.Action(actionType)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|-------------------|---|
| <i>actionType</i> | <p><code>LCACTION_RESET</code> – Returns the connector to a state equivalent to that just after connection. All outstanding results are committed and all result sets and state information are freed. Supported by all Connectors.</p> <p><code>LCACTION_TRUNCATE</code> – Deletes all records in the property <code>METADATA</code> in the most efficient available manner as determined by the connector.</p> <p><code>LCACTION_COMMIT</code> – Commits all changes in the current transaction. Only supported by Connectors with transaction functionality.</p> <p><code>LCACTION_ROLLBACK</code> – Rolls back all changes in the current transaction. Only supported by Connectors with transaction functionality.</p> <p><code>LCACTION_CLEAR</code> – Clears the current result set, freeing any locks, but does not affect any other context.</p> |

Example

```
Option Public
Option Explicit
Uselsx "*lsxlc" // DECS only, see overview in this chapter for
LEI syntax.

Sub Initialize

    Dim session As New LCSession

    Dim src As New LCConnection ("db2")
        //For LEI syntax, see overview.

    REM set properties to connect to both data sources

    src.Database = "Gold"

    src.Userid = "JDoe"

    src.Password = "xyzzzy"

    src.Metadata = "customer"

    REM now connect

    src.Connect

    ' check to see if the target metadata exists,
      if so clear it out.

    ' check for LCFAIL_INVALID_METADATA error
    On Error LCFAIL_INVALID_METADATA Goto NoMetadata
    Call src.Action (LCACTION_TRUNCATE)

    Print "the table '" & src.Metadata & "' has been truncated."
    Print "This removed all existing records."

    End

NoMetadata:

    ' couldn't truncate the table since it didn't exist

    Print "the table '" & src.Metadata & "' does not exist."

    End

End Sub
```

Example Output

```
the table 'customer' has been truncated.
This removed all existing records.
```

Call Method for LCConnection

This method is used to call a stored procedure and potentially produce a result set.

This method only supports input parameters to the stored procedure. If you want data returned from a stored procedure, it must be returned to a result set, not by output parameters.

Defined In
LCConnection

Syntax
count = *lcConnection.Call*(*parmFieldList*, *recordIndex*, *destFieldList*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|---|
| <i>parmFieldList</i> | LCFieldlist. The input parameter list for the stored procedure. |
| <i>recordIndex</i> | Long. The index location of the parameter values within the fieldlist. |
| <i>destFieldList</i> | LCFieldlist. Fieldlist to contain the metadata of the result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| <i>count</i> | The number of records affected by the call. Note that not all data sources return a <i>count</i> . LCCOUNT_OVERVIEW is returned if the count is not determined. |

Example

```
Option Public

Uselsx "lsxlc"
    //DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Con As New LCConnection ("sybase")
    //For LEI syntax, see overview.

Dim Parms As New LCFieldList

    Dim Result As New LCFieldList

    Dim Parm As LCField

    ' set properties to connect to both data sources
```

```

Con.Server = "Rainbow"
Con.Userid = "JDoe"
Con.Password = "xyzzzy"
' set the connection property to the stored procedure name
Con.Procedure = "NPInsertIntoSM_ADBOOK"
' now connect
Con.Connect
' append the new field to the fieldlist
Set Parm = Params.Append ("spParm", LCTYPE_TEXT)
' set the field to a value - in this case it is
' the one parameter needed for the stored procedure
Parm.text = " 'Edge' "
' using the fieldlist containing the field with the
' stored procedure parameter, call the stored procedure
If (Con.Call (Params, 1, Result) = 0) Then
    Print "No results were generated from the procedure call."
Else
    Print "A result set was generated from the procedure
call."
End If
End Sub

```

Example Output

A result set was generated from the procedure call.

Catalog Method for LCConnection

This method catalogs through metadata related information.

Any active result set for this connection will be replaced. Different metadata types may be cataloged, although all Connectors may not support all object types. The format of the result set produced is returned in the supplied fieldlist, and the result set contents can be retrieved with LCConnection.Fetch.

A connection is not required for Server and Database Catalogs for some Connectors. This is an exception, as a connection is required for all other result sets.

Defined In
LCCConnection

Syntax
count =lccConnection.Catalog(objectType, destFieldlist)

Parameters

| Parameter | Description |
|-------------------|---|
| <i>objectType</i> | <p>Long. Type of metadata information to be cataloged. Use an LCOBJECT_XXX constant to define the metadata type. The following list gives the required context and the resulting metadata format for each catalog type. The general fields, Name, Owner, and Comment, are produced for all objects, but only the first output field(name) is guaranteed to have data.</p> <p>All output fields are data type Text, unless otherwise specified. Field cataloging adds a fourth element, datatype.</p> <p>LCOBJECT_SERVER</p> <p><i>Context:</i> None</p> <p><i>Output Fields:</i> Server Name, Server Owner, Server Comment</p> <p>LCOBJECT_DATABASE</p> <p><i>Context:</i> SERVER property (if supported by this connector)</p> <p><i>Output Fields:</i> Database Name, Database Owner, Database Comment</p> <p>LCOBJECT_METADATA</p> <p><i>Context:</i> Current connection, ALTERNATE METADATA property</p> <p><i>Output Fields:</i> Metadata Name, Metadata Owner, Metadata Comment</p> <p>LCOBJECT_INDEX</p> <p><i>Context:</i> Current connection</p> <p><i>Output Fields:</i> Index Name, Index Owner, Index Comment</p> |

continued

| Parameter | Description |
|---|-------------|
| LCOBJECT_FIELD | |
| <i>Context:</i> Current connection, METADATA property, ALTERNATE METADATA property | |
| <i>Output Fields:</i> Field Name, Field Owner, Field Comment, Field Datatype Constant | |
| Value | Constant |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| <i>count</i> | Long. Number of catalog records available in the result set produced. This value is LCCOUNT_OVERVIEW if the number of records cannot be determined by the connector. |

Example

```
Option Public

Uselsx "xlsxc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim connect As New LCConnection ("db2")
    //For LEI syntax, see overview.

    Dim conFldLst As New LCFieldList

    Dim field As LCField

    ' this section assigns the appropriate properties to connect
    'to DB2
```

```

connect.Database = "Gold"
connect.Userid = "JDoe"
connect.Password = "xyzyz"
connect.Metadata = "customer"
' connect to DB2
connect.Connect
' now perform the catalog action - in this case for metadata
If (connect.Catalog (LCOBJECT_FIELD, conFldLst) = 0) Then
    Print "No tables were found."
Else
    ' fetch the results
    Set field = conFldLst.GetField(1)
    Print "The list of columns in the '" & _
        connect.Metadata & "' table include:"
    While (connect.Fetch (conFldLst) > 0)
        Print "      " & field.text(0)
    Wend
End If
End Sub

```

Example Output

The list of columns in the 'CUSTOMER' table include:

```

ACCOUNTMANAGER
CONTACTNAME
COMPANYNAME
COMPANYADDRESS
COMPANYCITY
COMPANYSTATE
COMPANYPHONE

```

Connect Method for LCConnection

This method establishes a connection to a connector. Multiple connections may be independently established to a single connector.

Defined In

LCConnection

Syntax

LcConnection.Connect

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim connect As New LCConnection ("db2")
    //For LEI syntax, see overview.

    ' set the appropriate properties to connect to DB2
    ' note the use of dynamic properties to do this
    ' all properties of a connection may be referenced
    ' by name

    connect.Database = "Gold"
    connect.Userid = "JDoe"
    connect.Password = "xyzzzy"

    REM try the connect

    connect.Connect

    Print "Successfully connected to DB2."

End Sub
```

Example Output

Successfully connected to DB2.

Copy Method for LCConnection

This method makes a copy of an existing connection, including all property values. Note that while all properties are copied, the current state of the connection and result set are not copied.

Defined In

LCConnection

Syntax

Set *newConnection* = *lcConnection*.Copy()

Parameters

lcConnection – The connection object that you want to copy.

Return Value

newConnection – The copy of the lcConnection object.

Example

```
Option Public

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim session As New LCSession

    Dim connect1 As New LCConnection ("db2")
        //For LEI syntax, see overview.

    Dim connect2 As LCConnection

    ' set the appropriate properties to connect to DB2
    ' note the use of dynamic properties to do this
    connect1.Database = "Gold"
    connect1.Userid = "JDoe"
    connect1.Password = "xyzzzy"

    ' now copy the connect
    Set connect2 = connect1.Copy

    If (connect2.Database = connect1.Database) Then
        Print "The copy of connection has the same database as the
            original."
    Else
```

```

        Print "The copy of connection has a different database
              from the original."

    End If

End Sub

Example Output
The copy of connection has the same database as the original.

```

Create Method for LCConnection

This method creates a metadata object. Each connector supports only certain object types. Refer to the documentation for the specific connector to determine the object types it supports.

Defined In
LCConnection

Syntax

Call *lcConnection.Create(objectType, srcFieldlist)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|-------------------|--|
| <i>objectType</i> | <p>Long. Type of object. Valid types and associated behavior include the following (refer to the documentation for the specific connector to determine which of the following are supported):</p> <p>LCOBJECT_SERVER Creates a server object (obtaining the name from the SERVER property). Additional information may be provided in Connector-specific properties.</p> <p>LCOBJECT_DATABASE Creates a database object (obtaining the name from the DATABASE property, and optionally the server name from the SERVER property). Additional information may be provided in Connector-specific properties.</p> <p>LCOBJECT_METADATA Creates a metadata object (obtaining the name from the METADATA property). The fields in the new metadata will have the same order, types, and names as fields in the fieldlist. Fields with the flag LCFIELDF_NO_CREATE are skipped.</p> |

continued

| <i>Parameter</i> | <i>Description</i> |
|---------------------|--|
| | <p>LCOBJECT_INDEX Creates an index object (the metadata being indexed is in METADATA property, the index name to create is in INDEX property). The key fields for the new index are fields in the fieldlist with the LCFIELD_KEY flag set.</p> <p>LCOBJECT_FIELD Creates a field object (the metadata being appended to is in METADATA property). The fields to append to the metadata will have the same order, types, and names as fields in the fieldlist. Fields with the flag LCFIELDF_NO_CREATE are skipped.</p> |
| <i>srcFieldlist</i> | LCFieldlist. Fieldlist from whose metadata or key fields the object is created. This parameter is ignored for object types SERVER and DATABASE. |

Example

```
Option Public
```

```
Option Explicit
```

```
Uselstx "*lsxlc"
```

```
    // DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim src As New LCCConnection ("db2")
```

```
        //For LEI syntax, see overview.
```

```
    Dim fldLstRecord As New LCFieldList
```

```
    Dim fld As LCField
```

```
    ' build the table definition
```

```
    ' note the use of the 'MaxLength' parameter
```

```
    ' this is used to more closely control the datatype creation
```

```
    ' within the connection - in this case DB2
```

```
    ' in DB2, a text stream with a MaxLength of 64 will be  
    ' created
```

```
    ' as VARCHAR(64). if the flag LCSTREAMF_FIXED was used
```

```
    ' the column would be CHAR(64). The field flag  
    ' LCFIELDF_NO_NULL
```

```
    ' would add NOT NULL to the column definition
```

```
    Call fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
```

```
    Set fld = fldLstRecord.Append ("CONTACTNAME", LCTYPE_TEXT)
```

```

Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = FldLstRecord.Append ("COMPANYNAME", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = FldLstRecord.Append ("COMPANYADDRESS",
    LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = FldLstRecord.Append ("COMPANYCITY", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = FldLstRecord.Append ("COMPANYSTATE", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 64, LCSTREAMFMT_LMBCS)
Set fld = FldLstRecord.Append ("COMPANYPHONE", LCTYPE_TEXT)
Call fld.SetFormatStream (0, 32, LCSTREAMFMT_LMBCS)
' set properties to connect to both data sources
src.Database = "Gold"
src.Userid = "JDoe"
src.Password = "xyzzzy"
src.Metadata = "customer"
src.Connect
' create it based on the metadata property already set above
On Error LCFAIL_DUPLICATE Goto tableexists
Call src.Create (LCOBJECT_METADATA, fldLstRecord)
Print "The '" & src.Metadata & "' table was created."
End
tableexists:
    Print "The '" & src.Metadata & "' table exists."
End
End Sub

```

Example Output

The 'customer' table was created.

Disconnect Method for LCCConnection

This method disconnects from a data source. Any existing result set is cleared.

Defined In

LCCConnection

Syntax

lcConnection.Disconnect

Example

Option Public

Usesxlx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim connect As New LCCConnection ("db2")

//For LEI syntax, see overview.

' set appropriate properties to connect to DB2

connect.Database = "Gold"

connect.Userid = "JDoe"

connect.Password = "DBINST1"

' connect to DB2 then disconnect

connect.Connect

Print "Successfully connected to DB2."

' now lets disconnect

connect.Disconnect

Print "Successfully disconnected from DB2."

End Sub

Example Output

Successfully connected to DB2.

Successfully disconnected from DB2.

Drop Method for LCConnection

This method drops the specified object type.

Each connector supports only certain object types. Refer to the documentation for the specific connector to determine the object types it supports.

Defined In
LCConnection

Syntax
Call *lcConnection.Drop(objectType)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|-------------------|---|
| <i>objectType</i> | Long. Type of object: Valid types and associated behavior are the following (refer to Appendix C for the specific connector to determine which of the following are supported): LCOBJECT_SERVER Drops a server object (obtaining the name from the SERVER property). LCOBJECT_DATABASE Drops a database object (obtaining the name from the DATABASE property, and optionally the server name from the SERVER property). LCOBJECT_METADATA Drops a metadata object (obtaining the name from the METADATA property). LCOBJECT_INDEX Drops an index object (the metadata indexed is in METADATA property, the index name to drop is in INDEX property). LCOBJECT_FIELD Drops a field object (metadata containing fields is in METADATA property, the fields being removed are in the FIELD_NAMES (or FieldNames) property). |

Example

```
Option Public

Option Explicit

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim src As New LCConnection ("db2")
    //For LEI syntax, see overview.
```

```

' set properties to connect to the data source

src.Database = "Gold"

src.Userid = "JDoe"

src.Password = "xyzyzy"

src.Metadata = "customer"

src.Connect

On Error Goto NoMetadata

Call src.Drop (LCOBJECT_METADATA)

Print "The '" & src.Metadata &
      "' table existed and had been deleted."

End

NoMetadata:

Print "The '" & src.Metadata & "' table did not exist."

End

End Sub

```

Example Output

The 'customer' table existed and had been deleted.

Execute Method for LCConnection

This method executes a statement in Connector-specific syntax, for example, an SQL statement for a relational database connector.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Execute**(*statement*, *destFieldlist*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|---|
| <i>statement</i> | The statement to execute in the syntax defined for the connector. See the documentation for the specific connector for information about the required syntax. |
| <i>destFieldList</i> | LCFieldlist. Fieldlist to contain the metadata of the execute result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing. |

Return Value

| | |
|--------------|---|
| <i>count</i> | Number of records selected or affected by this statement. If this number cannot be determined by the connector, the constant LCCOUNT_OVERVIEW is returned. |
|--------------|---|

Example

Option Public

Uselsx "*lsxlc"

 // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

 Dim src As New LCConnection ("db2")

 //For LEI syntax, see overview.

 Dim fldLst As New LCFieldList

 Dim fld As LCField

 Dim count As Integer

 ' set the appropriate properties to connect

 src.Database = "Gold"

 src.Userid = "JDoe"

 src.Password = "xyzzzy"

 src.Connect

 ' now connected, we can execute a selection statement

 If (src.Execute ("SELECT * from customer", fldLst) = 0)

Then

 Print "No records were fetched."

 End

End If

 Set fld = fldLst.Lookup ("CONTACTNAME")

 Print "the 'contact names' stored in the table are:"

 ' fetch each record from the result set

 While (src.Fetch (fldLst) > 0)

 count = count + 1

 Print " record #" & Cstr(count) & " =
 '" & fld.text(0) & "'"

 Wend

```
    If (count = 0) Then Print "No records were fetched."  
End Sub
```

Example Output

the 'contact names' stored in the table are:

```
record #1 = 'Peter Pan'  
record #2 = 'R. U. Happy'  
record #3 = 'Issac Bernard Mathews'
```

Fetch Method for LCCConnection

This method obtains the next group of records from a result set. This method requires an active result set in the Connection.

Defined In

LCCConnection

Syntax

count = *lcConnection*.Fetch(*destFieldlist*, *recordIndex*, *recordCount*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|--|
| <i>destFieldlist</i> | LCFieldlist. Fieldlist to receive the data. For each field in Fieldlist without the flag LCFIELDF_NO_FETCH, data from the corresponding field in the result set will be copied into that field. Fields in the result set and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise, and are type-checked before retrieving data. |
| <i>recordIndex</i> | Long. Optional. Starting record index in the fieldlist where the record will be stored. Default is 1. |
| <i>recordCount</i> | Long. Optional. Number of records to fetch. The number of records actually fetched may be less than this number if the end of the result set was reached. While all Connectors can fetch multiple records, only Connectors which indicate support for Array Fetch perform a true multi-record fetch and therefore reduce network traffic and increase performance. Default is 1. |

Return Value

count Long. Number of records successfully fetched.

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive fetches from the same target.

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim src As New LCConnection ("db2")

//For LEI syntax, see overview.

Dim fldLst As New LCFieldList

Dim keyLst As New LCFieldList

Dim fld As LCField

Dim count As Integer

' set the appropriate properties to connect to the data
'source

src.Database = "Gold"

src.Userid = "JDoe"

src.Password = "xyzzzy"

src.Metadata = "customer"

src.Connect

' the FIELDNAMES property of a connection is used to

' specify which fields should be used in the resultset

' if no names are listed, then all fields will be fetched

src.FieldNames = "ContactName, AccountManager"

' the select statement may be called with 'Nothing' as

' the keylist parameter. this causes all records to be

' selected for the result set.

' by creating a keylist with one or more keys, conditions,

' and values, tighter control of the result set is possible

' here we want to indicate all account managers except

' number 200

' NOTE: to indicate that a field is a key, the LCFIELD_KEY
flag

' must always be included in the value of the connection's
flags

```

Set fld = keyLst.Append ("ACCOUNTMANAGER", LCTYPE_INT)
fld.Flags = LCFIELD_KEY_NE Or LCFIELD_KEY
fld.Value = 200
' the selection statement builds an international result set
  which
' is later accessed with successive fetches
If (src.Select (keyLst, 1, fldLst) = 0) Then
    Print "No data were located."
    End
End If
Set fld = fldLst.Lookup ("CONTACTNAME")
Print "the 'contact names' stored in the table are:"
REM fetch a record from the result set
While (src.Fetch (fldLst) > 0)
    count = count + 1
    Print "        record #" & Cstr(count) & " =
        '" & fld.text(0) & '"
Wend
If (count = 0) Then Print "The table contains no records."
End Sub

```

Example Output

the 'contact names' stored in the table are:

```

record #1 = 'Peter Pan'
record #2 = 'R. U. Happy'
record #3 = 'Issac Bernard Mathews'

```

GetProperty Method for LCCConnection

This method retrieves a copy of the current value for a connection property. Note that use of dynamic properties is more efficient.

Defined In

LCCConnection

Syntax

Call thisConnection.GetProperty(propertyToken, destField)

Parameters

| Parameter | Description |
|----------------------|---|
| <i>propertyToken</i> | Long. Token representing the connector property for which the value is returned. See Appendix B for a list of tokens. |
| <i>destField</i> | Current value for the connector property. If the property value is of a different type than this field, then data conversion will occur, if possible. The property value will be written into the first value in the field. |

Example

```
Option Public
Option Explicit
Uselsx "xlsxl"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim connect As New LCConnection ("db2")
    //For LEI syntax, see overview.
    Dim conFld As LCField
    ' get the value of the server property
    Set conFld = connect.GetProperty (LCTOKEN_WRITEBACK)
    Print "The writeback property value is: " & conFld.text(0)
End Sub
```

Example Output

The writeback property value is: 0

GetProperty<Type> Methods

This method returns a connector property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

This method allows retrieval of Connect properties as with LCConnection.GetProperty, but does not require a field object. If the property value is of a different type, data conversion will be performed. The value of the password property cannot be obtained with this function, and will result in an INVALID_PROPERTY error.

Defined In

LCCConnection

Syntax

```
flag = lcConnection.GetPropertyBoolean(propertyToken, default)  
Set destCurrency = lcConnection.GetPropertyCurrency(propertyToken)  
Set destDatetime = lcConnection.GetPropertyDatetime(propertyToken)  
destFloat = lcConnection.GetPropertyFloat(propertyToken)  
destInt = lcConnection.GetPropertyInt(propertyToken)  
Set destNumeric = lcConnection.GetPropertyNumeric(propertyToken)  
Set destStream = lcConnection.GetPropertyStream(propertyToken,  
streamFormat)
```

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|--|
| <i>propertyToken</i> | A token identifying the Connection property. See Appendix B for a list of property tokens. |
| <i>default</i> | (GetPropertyBoolean only) Value to be returned if the property cannot be located. Default value is FALSE. |
| <i>streamFormat</i> | (GetPropertyStream only) Format that the stream is converted to before being returned. If <i>streamFormat</i> is zero, no conversion occurs and the stream is copied in the same format as the property value. |

Return Values

| | |
|-------------------------|--|
| <i>flag</i> | GetPropertyBoolean only. Boolean value, either TRUE or FALSE. If the property does not exist, the default is returned. |
| <i>dest<Type></i> | Current value for the connector property. |

Examples

Option Public

```
Uselxx "xlslc" // DECS only, see overview in this chapter for  
LEI syntax.
```

Sub Initialize

```
Dim connect As New LCCConnection ("oracle") //For LEI syntax,  
see overview.
```

```
Dim conFld As LCField
```

```
Dim propName As String
```

```

Dim propDate As LCDateTime
Dim propNumeric As LCNumeric
Dim propStrm As LCStream
Dim propCurr As LCCurrency
Dim propFloat As Double
Dim propInt As Long
Dim propBool As Variant
Dim tokenId As Long
Dim propType As Long
Dim propFlags As Long
' set some connector properties
connect.Server = "Rainbow"
connect.Userid = "JDoe"
connect.Password = "xyzzzy"
' it is not necessary to connect to list properties
Call connect.ListProperty (LCLIST_FIRST, _
tokenId, propType, propFlags, propName)
Print "NAME" Tab(20); "ID"; Tab(28); "FLAGS"; _
Tab(36); "TYPE"; Tab(48); "VALUE"
Print "-----" Tab(20); "-----"; Tab(28);
      "-----"; _
Tab(36); "-----"; Tab(48); "-----"
Do
    Set conFld = connect.GetProperty (tokenId)
    ' match the property to a datatype and fetch it as that
      datatype
    Select Case propType
    Case LCTYPE_DATETIME:
        Set propDate = connect.GetPropertyDatetime (tokenId)
        Print propName; Tab(20); Hex(tokenId); Tab(28);
          Hex(propFlags); _
        Tab(36); "LCDatetime"; Tab(48); propDate.text

```

```

Case LCTYPE_NUMERIC:
    Set propNumeric = connect.GetPropertyNumeric (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(28);
        Hex(propFlags); _
    Tab(36); "LCNumeric"; Tab(48); propNumeric.text
Case LCTYPE_TEXT:
    Set propStrm = connect.GetPropertyStream (tokenId,
        LCSTREAMFMT_NATIVE)
    Print propName; Tab(20); Hex(tokenId); Tab(28);
        Hex(propFlags); _
    Tab(36); "LCStream"; Tab(48); propStrm.text
Case LCTYPE_CURRENCY:
    Set propCurr = connect.GetPropertyCurrency (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(28);
        Hex(propFlags); _
    Tab(36); "LCCurrency"; Tab(48); propCurr.text
Case LCTYPE_FLOAT:
    propFloat = connect.GetPropertyFloat (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(28);
        Hex(propFlags); _
    Tab(36); "Double"; Tab(48); Cstr(propFloat)
Case LCTYPE_INT:
    If (propFlags And LCPROPERTYF_BOOLEAN) Then
        propBool = connect.GetPropertyBoolean (tokenId, False)
        Print propName; Tab(20); Hex(tokenId); Tab(28);
            Hex(propFlags); _
        Tab(36); "Boolean"; Tab(48); Cstr(propBool)
    Else
        propInt = connect.GetPropertyInt (tokenId)
        Print propName; Tab(20); Hex(tokenId); Tab(28);
            Hex(propFlags); _
        Tab(36); "Long"; Tab(48); Cstr(propInt)
    End If
End Select

```

```

    Loop _
    While connect.ListProperty (LCLIST_NEXT, _
    tokenId, propType, propFlags, propName)
End Sub

```

Example Output

| NAME | ID | FLAGS | TYPE | VALUE |
|-------------------|-------|-------|------------|-----------|
| ----- | ----- | ----- | ----- | ----- |
| Name | 30004 | 4 | LCStream | oracle |
| IsConnected | 3000C | 6 | Boolean | False |
| Server | 10001 | 1 | LCStream | mycyclone |
| Userid | 10003 | 1 | LCStream | scott |
| Metadata | 10005 | 0 | LCStream | |
| Index | 10006 | 0 | LCStream | |
| MapByName | 10007 | 2 | Boolean | False |
| Writeback | 10008 | 2 | Boolean | False |
| Condition | 1000B | 0 | LCStream | |
| StampField | 1000C | 0 | LCStream | |
| BaseStamp | 1000D | 0 | LCDatetime | |
| MaxStamp | 1000E | 0 | LCDatetime | |
| TextFormat | 1000F | 4 | Long | 65535 |
| CharacterSet | 30008 | 4 | LCStream | NATIVE |
| Procedure | 10010 | 0 | LCStream | |
| Owner | 10011 | 0 | LCStream | |
| AlternateMetadata | 10013 | 2 | Boolean | False |
| CommitFrequency | 1 | 0 | Long | 0 |
| RollbackOnError | 2 | 2 | Boolean | False |
| CreateLongColumn | 3 | 0 | LCStream | |
| CreateLongByUser | 4 | 0 | Long | 0 |
| TraceSQL | 5 | 2 | Boolean | False |

Insert Method for LCConnection

This method inserts a specified number of records into the connection metadata.

Note that you can achieve optimal performance by using the same fieldlist across consecutive inserts from the same target.

Defined In
LCConnection

Syntax
count = *lcConnection.Insert(srcFieldlist, recordIndex, recordCount)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|---------------------|--|
| <i>srcFieldlist</i> | LCFieldlist. Fieldlist containing the records to insert. For each field in Fieldlist without the flag LCFIELDF_NO_INSERT, data from the corresponding field in the result set will be copied into that field. Fields in the result set and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise, and are type-checked before inserting data. |
| <i>recordIndex</i> | Long. Optional. Starting record index of the insert in the fieldlist. The default is 1. |
| <i>recordCount</i> | Long. Optional. Number of records to insert. The number of records actually inserted may be less than this number if an error was encountered. While all Connectors can insert multiple records, only Connectors that indicate support for Array Insert perform a true multi-record insert and therefore reduce network traffic and increase performance. The default is 1. |

Return Value

| | |
|--------------|--|
| <i>count</i> | Long. Number of records successfully inserted. |
|--------------|--|

Example

```
Option Public
Uselsx "1sxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim src As New LCConnection ("db2")
    //For LEI syntax, see overview.
    Dim fields As New LCFieldList (5)
    Dim field As LCField
```

```

Dim data(4) As String
Dim idata(4) As Long

REM set the appropriate properties to connect to the data
sources

src.Database = "Gold"
src.Userid = "JDoe"
src.Password = "xyzzzy"
src.Metadata = "customer"
src.MapByName = True

REM connect to the two data sources
src.Connect

REM use a key to find certain records to remove
Set field = fields.Append ("ACCOUNTMANAGER", LCTYPE_INT)
idata(0) = 100
idata(1) = 200
idata(2) = 300
idata(3) = 400
idata(4) = 200

field.value = idata

Set field = fields.Append ("CONTACTNAME", LCTYPE_TEXT)
data(0) = "Peter Pan"
data(1) = "Big Steel"
data(2) = "R. U. Happy"
data(3) = "Issac Bernard Mathews"
data(4) = "Paula Falderall"

field.value = data

Set field = fields.append ("COMPANYADDRESS", LCTYPE_TEXT)
data(0) = "One Bit Tree"
data(1) = "Gurder Way"
data(2) = "Daisy Hill Pup Farm"
data(3) = "Big Blue Ave."
data(4) = "Planet Hollywood"

```

```

    field.value = data

    Set field = fields.Append ("COMPANYSITY", LCTYPE_TEXT)

    data(0) = "Never Never"
    data(1) = "Iron"
    data(2) = "Beagle"
    data(3) = "New York"
    data(4) = "Parthenon"

    field.value = data

    Set field = fields.Append ("COMPANYSSTATE", LCTYPE_TEXT)

    data(0) = "Land"
    data(1) = "PA"
    data(2) = "WI"
    data(3) = "NY"
    data(4) = "AQ"

    field.value = data

    REM we can perform a keyed delete of all matching records in
    the table

    Print "Inserted " & Cstr (src.Insert (fields, 1, 5)) & "
    record(s)."

End Sub

```

Example Output

```
Inserted 5 record(s).
```

ListProperty Method for LCCConnection

This method obtains the first or next property information for properties supported by this connector.

Defined In

LCCConnection

Syntax

Call *lcConnection*.**listProperty** (*list*, *propertyToken*, *dataType*, *propertyFlags*, *propertyName*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|--|
| <i>list</i> | Long. Constant indicating whether to return the first or next connector property. LCLIST_FIRSTReturn the first property in the property list. LCLIST_NEXTReturn the next property (or the first property if this is the first call to this function for this Connection). |
| <i>propertyToken</i> | Long. Optional. Token assigned to the property. |
| <i>dataType</i> | Long. Optional. Data type of the property. |
| <i>propertyFlags</i> | Long. Optional. Property flags for the property; one or more of the flags below Ored together. LCPROPERTYF_CONNECTProperty is used for connecting LCPROPERTYF_BOOLEANProperty is a boolean value LCPROPERTYF_READONLYProperty is read-only LCPROPERTYF_TEXTLISTProperty is a text list |
| <i>PropertyName</i> | String. Optional. Name of the property. |

Example

Option Public

Usesx "1sxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim connect As New LCConnection ("oracle")

//For LEI syntax, see overview.

Dim conFld As LCField

Dim propName As String

Dim tokenId As Long

Dim propType As Long

Dim propFlags As Long

' set some connector properties

connect.Server = "Rainbow"

connect.Userid = "JDoe"

connect.Password = "xyzyz"

connect.Metadata = "scott.bigtable"

connect.FieldNames = "name, address, city, state, zipcode,
phone"


```

Print "NAME" Tab(20); "ID" Tab(26); "FLAGS"; _
Tab(32); "TYPE"; Tab(38); "VALUE"
Print "-----" Tab(20); "-----"; Tab(26);
    "-----"; _
Tab(32); "-----"; Tab(38); "-----"
' all of the parameters are optional and any may be omitted
Call connect.ListProperty (LCLIST_FIRST, _
tokenId, propType, propFlags, propName)
Do
    Set conFld = connect.GetProperty (tokenId)
    Print propName; Tab(20); Hex(tokenId); Tab(27);
        Hex(propFlags); _
    Tab(32); propType; Tab(38); conFld.Text(0)
    Loop While connect.ListProperty (LCLIST_NEXT, _
tokenId, propType, propFlags, propName)
End Sub

```

Example Output

| NAME | ID | FLAGS | TYPE | VALUE |
|-------------|-------|-------|------|---|
| ----- | ----- | ----- | ---- | ----- |
| Name | 30004 | 4 | 6 | oracle |
| IsConnected | 3000C | 6 | 1 | 0 |
| Server | 10001 | 1 | 6 | mycyclone |
| Userid | 10003 | 1 | 6 | scott |
| Password | 10004 | 1 | 7 | |
| Metadata | 10005 | 0 | 6 | scott.bigtable |
| Index | 10006 | 0 | 6 | |
| MapByName | 10007 | 2 | 1 | 0 |
| Writeback | 10008 | 2 | 1 | 0 |
| OrderNames | 1000A | 8 | 7 | |
| FieldNames | 10009 | 8 | 7 | name, address, city, state, zipcode, phone |
| Condition | 1000B | 0 | 6 | |
| StampField | 1000C | 0 | 6 | |

| | | | | |
|-------------------|-------|---|---|--------|
| BaseStamp | 1000D | 0 | 5 | |
| MaxStamp | 1000E | 0 | 5 | |
| TextFormat | 1000F | 4 | 1 | 65535 |
| CharacterSet | 30008 | 4 | 6 | NATIVE |
| Procedure | 10010 | 0 | 6 | |
| Owner | 10011 | 0 | 6 | |
| AlternateMetadata | 10013 | 2 | 1 | 0 |
| CommitFrequency | 1 | 0 | 1 | 0 |
| RollbackOnError | 2 | 2 | 1 | 0 |
| CreateLongColumn | 3 | 0 | 6 | |
| CreateLongByUser | 4 | 0 | 1 | 0 |
| TraceSQL | 5 | 2 | 1 | 0 |

LookupProperty Method for LCConnection

This method determines if a connector supports a specified property.

Defined In
LCConnection

Syntax

flag = lcConnection.LookupProperty(propertyToken, dataType, propertyFlags, propertyName)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|---|
| <i>propertyToken</i> | A token identifying the Connection property. See Appendix B for a list of property tokens. |
| <i>dataType</i> | Long. Optional. The property data type. |
| <i>propertyFlags</i> | Long. Optional. The property flags from the following list. LCPROPERTYF_CONNECTProperty is used for connecting. LCPROPERTYF_BOOLEANProperty is a boolean value. LCPROPERTYF_READONLYProperty is read-only. LCPROPERTYF_TEXTLISTProperty is a text list. |
| <i>propertyName</i> | String. Optional. The name of the property. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| flag | TRUE or FALSE, indicating whether the property is supported for this connection. |

Example

Option Public

```
Uselsx *lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
Dim connect As New LCConnection ("oracle")
```

```
//For LEI syntax, see overview.
```

```
REM note that a connect is not necessary to list and look up  
properties
```

```
REM the parameters are optional, to text for the existence  
of a property
```

```
REM simply provide the TOKEN parameter
```

```
If (connect.LookupProperty (LCTOKEN_SERVER)) Then
```

```
Print "Oracle has a 'Server' property"
```

```
Else
```

```
Print "Oracle does not have a 'Server' property"
```

```
End If
```

```
If (connect.LookupProperty (LCTOKEN_DATABASE)) Then
```

```
Print "Oracle has a 'Database' property"
```

```
Else
```

```
Print "Oracle does not have a 'Database' property"
```

```
End If
```

```
End Sub
```

Example Output

```
Oracle has a 'Server' property
```

```
Oracle does not have a 'Database' property
```

Remove Method for LCCConnection

This method performs either a writeback result set remove or a keyed remove of records in the external system.

Defined In
LCCConnection

Syntax
count = *lcConnection.Remove*(*keyFieldlist*, *recordIndex*, *recordCount*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|---------------------|--|
| <i>keyFieldlist</i> | LCFieldlist. Fieldlist containing keys for a keyed remove only. For each field in the Fieldlist with the flag LCFIELDF_KEY set, only records in the connection with the same value for that field are deleted. Use LCFIELDF_KEY_XXX flags as inequality keys, if desired. Zero or more key fields may be supplied, with zero keys resulting in all records in the target being deleted. Fields in the connection and fieldlist are matched by name if the MapByName property is TRUE, by position otherwise. |
| <i>recordIndex</i> | Long. Optional. Starting record index in the fieldlist. The default is 1. |
| <i>recordCount</i> | Long. Optional. Number of keyed remove operations to perform. RecordCount must be 1 for a writeback remove but may be greater to perform multiple keyed updates with different update and key values. The default is 1. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| <i>count</i> | Long. Number of records successfully removed. If this number cannot be determined, the constant LCCOUNT_OVERVIEW is returned. |

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive removes from the same target.

The property Writeback indicates whether to perform a writeback or keyed remove:

- Writeback remove: If the Writeback property is now set, this method removes the most recently fetched record from the writeback result set (the result set produced by LCCConnection.Execute, LCCConnection.Select or LCCConnection.Call with the Writeback property set).

- **Keyed remove:** Removes all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELDF_KEY field flag set. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant connector. See the documentation for the connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELDF_KEY
- greater than or equal to LCFIELDF_KEY + LCFIELDF_KEY_GT
- less than or equal to LCFIELDF_KEY + LCFIELDF_KEY_LT
- not equal to LCFIELDF_KEY + LCFIELDF_KEY_NE
- greater than LCFIELDF_KEY + LCFIELDF_KEY_GT + LCFIELDF_KEY_NE
- less than LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim src As New LCConnection ("db2")

//For LEI syntax, see overview.

Dim keyList As New LCFieldList

Dim key As LCField

REM set the appropriate properties to connect to the data sources

src.Database = "Gold"

src.Userid = "JDoe"

src.Password = "xyzzzy"

src.Metadata = "customer"

src.MapByName = True

REM connect to the two data sources

src.Connect

REM use a key to find certain records to remove

```

Set key = keyList.Append ("ACCOUNTMANAGER", LCTYPE_INT)
key.Flags = LCFIELD_KEY
key.value = 200

REM we can perform a keyed delete of the matching record in
the table

Print "Removed " & Cstr (src.Remove (keyList)) & "
record(s) "

End Sub

```

Example Output

Removed 2 record(s)

Select Method for LCConnection

This method produces a result set from the current METADATA property and other properties.

Defined In

LCConnection

Syntax

count = *lcConnection*.**Select** (*keyFieldlist*, *recordIndex*, *destFieldlist*)

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|---|
| <i>keyFieldlist</i> | LCFieldlist. Selection keys. All fields in KeyFieldlist with the LCFIELD_KEY flag set are used as the selection keys. Only records in the connection with the same value for key fields will be selected. Additional LCFIELD_KEY_XXX flags (GT – greater than, LT – less than, NE – not equal) allow inequality keys to be used. Zero or more key fields may be supplied to restrict the result set. No keys or Nothing will select all records for the result set. |
| <i>recordIndex</i> | Long. Record index position in the key fieldlist from which to obtain the record containing key field values. |
| <i>destFieldlist</i> | LCFieldlist. Fieldlist to contain the metadata of the selected result set. The fields in the result set will be appended to this fieldlist. If the result set metadata is not required, use Nothing. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| <i>count</i> | Long. Number of records in the result set. If this number cannot be determined by the connector, the constant LCCOUNT_OVERVIEW is returned. |

Usage Notes

The property Writeback indicates whether to perform a writeback or keyed selection:

- Writeback selection: If the Writeback property is now set, this method selects fields in the most recently fetched record from the writeback result set (the result set produced by LCCConnection.Execute, LCCConnection.Select or LCCConnection.Call with the Writeback property set).
- Keyed selection: Selects all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELDF_KEY field flag set. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant connector. See the documentation for the connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELDF_KEY
- greater than or equal to LCFIELDF_KEY + CFIELDF_KEY_GT
- less than or equal to LCFIELDF_KEY + LCFIELDF_KEY_LT
- not equal to LCFIELDF_KEY + LCFIELDF_KEY_NE
- greater than LCFIELDF_KEY + LCFIELDF_KEY_GT + LCFIELDF_KEY_NE
- less than LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

```
Dim src As New LCCConnection ("db2")
//For LEI syntax, see overview.
```

```
Dim fldLst As New LCFieldList
```

```
Dim fld As LCField
```

```
Dim count As Integer
```

```
REM set the appropriate properties to connect to the data
sources
```

```
src.Database = "Gold"
```

```

src.Userid = "JDoe"
src.Password = "xyzzzy"
src.Metadata = "customer"
REM connect to the two data sources
src.Connect
REM now connected, we can execute a selection statement
count = src.Select (Nothing, 1, fldLst)
Select Case count
Case LCCOUNT_OVERVIEW
    Print "An overview number of records were located."
Case 0
    Print "No data were located."
Case Else
    Print "The table contains " & Cstr(count) & " records."
End Select
End Sub

```

Example Output

An overview number of records were located.

SetProperty Method for LCConnection

This method assigns the value for a connection property. Note that use of dynamic properties is more efficient.

Defined In

LCConnection

Syntax

Call *thisConnection.SetProperty(propertyToken, srcField)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|----------------------|--|
| <i>propertyToken</i> | Long. Token representing the connector property for which the value is to be assigned. See Appendix B for a list of tokens. |
| <i>srcField</i> | LCField. New value for the connector property. If the property value is of a different type than this field, then data conversion will occur. The property value will be assigned from the first value in the field. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim connect As New LCConnection ("oracle")
    //For LEI syntax, see overview.

    Dim conFld As New LCField (LCTYPE_TEXT, 1)

    REM set appropriate properties to connect to oracle
    conFld.text = "Barker"

    Call connect.SetProperty (LCTOKEN_SERVER, conFld)

    REM example of using the expanded property

    Print "Here is the server property value: " & connect.Server

End Sub
```

Example Output

Here is the server property value: Barker

SetProperty<Type> methods for LCConnection

These methods assign a Connection property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

Defined In

LCConnection

Syntax

Call *lcConnection.SetPropertyBoolean(propertyToken, srcBoolean)*

Call *lcConnection.SetPropertyCurrency(propertyToken, srcCurrency)*

Call *lcConnection.SetPropertyDatetime(propertyToken, srcDatetime)*

Call *lcConnection.SetPropertyFloat(propertyToken, srcFloat)*

Call *lcConnection.SetPropertyInt(propertyToken, srcInt)*

Call *lcConnection.SetPropertyNumeric(propertyToken, srcNumeric)*

Call *lcConnection.SetPropertyStream(propertyToken, srcStream)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|------------------------|---|
| <i>propertyToken</i> | Long. Token identifying the Connection property. See Appendix B for a list of tokens. |
| <i>src<Type></i> | Value to assign to the Connection property. |

Example

Option Public

```
Uselx "xlsxl"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim connect As New LCConnection ("db2")
```

```
    //For LEI syntax, see overview.
```

```
    Dim conFld As New LCField (LCTYPE_TEXT,1)
```

```
    Dim propName As String
```

```
    Dim propDate As New LCDateTime
```

```
    Dim propNumeric As New LCNumeric
```

```
    Dim propStrm As New LCStream (0, 0, LCTYPE_TEXT)
```

```
    Dim propCurr As New LCCurrency
```

```
    Dim propFloat As Double
```

```
    Dim propInt As Long
```

```
    Dim tokenId As Long
```

```
    Dim propType As Long
```

```
    Dim propFlags As Long
```

```
    ' set the appropriate properties to connect to DB2,
```

```
    ' note that you can also use dynamic properties to do this
```

```
    connect.Database = "Gold"
```

```
    connect.Userid = "JDoe"
```

```
    connect.Password = "xyzyz"
```

```
    Call connect.ListProperty (LCLIST_FIRST, _
```

```
    tokenId, propType, propFlags, propName)
```

```
    Do
```

```
        Set conFld = connect.GetProperty (tokenId)
```

```

If (propFlags And LCPROPERTYF_READONLY) <>
    LCPROPERTYF_READONLY Then

    ' match the property to a datatype and set it

Select Case propType
Case LCTYPE_DATETIME:
    propDate.SetCurrent
    Call connect.SetPropertyDatetime (tokenId, propDate)
    Print propName & " is now a datetime and contains " &
propDate.text

Case LCTYPE_NUMERIC:
    propNumeric.text = "100.0123456789"
    Call connect.SetPropertyNumeric (tokenId, propNumeric)
    Print propName & " is now a numeric and contains
        " & propNumeric.text

Case LCTYPE_TEXT:
    propStrm.text = "a beautiful day"
    Call connect.SetPropertyStream (tokenId, propStrm)
    Print propName & " is now text and contains
        " & propStrm.text

Case LCTYPE_CURRENCY:
    propCurr.text = "140.10"
    Call connect.SetPropertyCurrency (tokenId, propCurr)
    Print propName & " is now currency and contains
        " & propCurr.text

Case LCTYPE_FLOAT:
    Call connect.SetPropertyFloat (tokenId, 30000.456)
    Print propName & " is now a float and contains
        " & Cstr (30000.456)

Case LCTYPE_INT:
    If (propFlags And LCPROPERTYF_BOOLEAN) Then
        Call connect.SetPropertyBoolean (tokenId, True)
        Print propName & " is now an int and contains
            " & Cstr(True)
    Else

```

```

        Call connect.SetPropertyInt (tokenId, 123)

        Print propName & " is now an int and contains
            " & Cstr(123)

    End If

End Select

End If

Loop _

While connect.ListProperty (LCLIST_NEXT, _
    tokenId, propType, propFlags, propName)

End Sub

```

Example Ouput

```

Database is now text and contains a beautiful day

Userid is now text and contains a beautiful day

Metadata is now text and contains a beautiful day

Index is now text and contains a beautiful day

MapByName is now an int and contains True

Writeback is now an int and contains True

Condition is now text and contains a beautiful day

StampField is now text and contains a beautiful day

BaseStamp is now a datetime and contains 09/08/1998
    05:24:33.65 PM

MaxStamp is now a datetime and contains 09/08/1998
    05:24:33.65 PM

Procedure is now text and contains a beautiful day

Owner is now text and contains a beautiful day

AlternateMetadata is now an int and contains True

CommitFrequency is now an int and contains 123

RollbackOnError is now an int and contains True

CreateMaxLogged is now an int and contains 123

NoJournal is now an int and contains True

CreateInDatabase is now text and contains a beautiful day

TraceSQL is now an int and contains True

```

Update Method for LCConnection

This method updates selected records in the connection metadata.

Defined In
LCConnection

Syntax
count = *lcConnection.Update(srcFieldlist, recordIndex, recordCount)*

Parameters

| <i>Parameter</i> | <i>Description</i> |
|---------------------|--|
| <i>srcFieldlist</i> | LCFieldlist. The fieldlist that contains the fields to be changed. |
| <i>recordIndex</i> | Long. Optional. The starting record in the fieldlist. The default is 1. |
| <i>recordCount</i> | Long. Optional. The number of records in the fieldlist to use to perform the update. The default is 1. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| <i>count</i> | Long. Number of records successfully updated. This may be LCCOUNT_OVERVIEW. |

Usage Notes

You can achieve optimal performance by using the same fieldlist across consecutive updates to the same target.

The property Writeback indicates whether to perform a writeback or keyed update:

- Writeback update: If the Writeback property is now set, this method updates fields in the most recently fetched record from the writeback result set (the result set produced by LCConnection.Execute, LCConnection.Select or LCConnection.Call with the Writeback property set). Fields that have the NO_UPDATE field flag set are not updated.
- Keyed update: Updates all records in the supplied metadata with field values matching all fields in the fieldlist that have the LCFIELD_KEY field flag set. Fields with the NO_UPDATE field flags set are not affected. If a value is specified for the property Condition, this will be included in the key search criteria. This value must be in a syntax valid for the relevant connector. See the documentation for the connector for information about its supported syntax.

When using inequality key flags GT, LT, and NE, it is important to remember that the default of no flags is equal. The following combinations are valid for inequality flags:

- equal to LCFIELDF_KEY
- greater than or equal to LCFIELDF_KEY + LCFIELDF_KEY_GT
- less than or equal to LCFIELDF_KEY + LCFIELDF_KEY_LT
- not equal to LCFIELDF_KEY + LCFIELDF_KEY_NE
- greater than LCFIELDF_KEY + LCFIELDF_KEY_GT + LCFIELDF_KEY_NE
- less than LCFIELDF_KEY + LCFIELDF_KEY_LT + LCFIELDF_KEY_NE

Example

Option Public

Option Explicit

Uselsx "xlsxl" 'c

 // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

 Dim src As New LCConnection ("db2")

 //For LEI syntax, see overview.

 Dim fldList As New LCFieldList

 Dim fld As LCField

 ' set the appropriate properties to connect to the data source

 src.Database = "Gold"

 src.Userid = "JDoe"

 src.Password = "xyzzzy"

 src.Metadata = "customer"

 src.Connect

 ' use a key to find certain records to update

 Set fld = fldList.Append ("ACCOUNTMANAGER", LCTYPE_INT)

 fld.Flags = LCFIELDF_KEY

 fld.value = 200

 ' set the field which will be changed, and set the new value

 Set fld = fldList.Append ("CONTACTNAME", LCTYPE_TEXT)

 fld.text = "Me"

```
src.MapbyName = True  
  
' set the contact's city to "Denver"  
  
' for the record who's contact is "Me"  
  
Print "The update affected " & Cstr (src.Update (fldList))  
    & " records"  
  
End Sub
```

Example Output

The update affected 2 records.

Chapter 3

LCCurrency Class

This chapter provides information about the Lotus Connectors LCCurrency class methods and properties

Overview

The LCCurrency class represents a currency value and has the same format and restrictions as the LotusScript currency datatype. The value is an 8-byte integer with a fixed scale of 4 decimal places, providing 19 digits of precision. Currency is commonly used when higher precision is required, such as for monetary amounts. A currency is much more precise than an integer, more precise than a float, and more efficient (but less precise) than a numeric. Note that during any currency overflow, the maximum or minimum valid currency value is assigned in addition to the error generated.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session (this becomes the default Log Document name)
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "*lsxlei"

Sub Initialize

    Dim MyLEISession As New LCSession ("MainSession")

    Dim MyNotesConnection as New LCConnection
        ("MyNotesConn")

    Dim MyOracleConnection as New LCConnection
        ("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

Type CURRENCY Format and Values

The following table shows CURRENCY format and values.

| <i>Description</i> | <i>Value</i> |
|--------------------|---|
| Type constant | LCTYPE_CURRENCY |
| Description | 8-byte integer with a fixed scale of 4 |
| Other | Precision (LCMAX_CURRENCY_PREC) = 19 Scale (LCMAX_CURRENCY_SCALE) = 4 Minimum Value (LCMIN_CURRENCY_VALUE) = -922,337,203,685,477.5807 Maximum Value (LCMAX_CURRENCY_VALUE) = 922,337,203,685,477.5807 |

LCCurrency Class Methods Summary

The following are the LCCurrency class methods:

| <i>Value</i> | <i>Description</i> |
|---------------------|--|
| LCCurrency.Add | Adds two currency values and deposits the result in the object making the call. |
| LCCurrency.Compare | Compares two currency values returning a value indicating the relationship between them. |
| LCCurrency.Copy | Makes a copy of a currency. |
| LCCurrency.Subtract | Subtracts one currency value from another, and returns the result to the object making the call. |

LCCurrency Properties

LCCurrency has two properties: Text and Value.

- Text
Text is a string representation.
- Value
Value is of LotusScript currency data type.

New Method for LCCurrency

This is the constructor for LCCurrency. It initializes a currency object.

Defined In

LCCurrency

Syntax

Dim variableName as New LCCurrency

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim num1 As New LCCurrency
```

Add Method for LCCurrency

This method adds two LCCurrency values, producing the sum. The sum is returned to the object that is making the call.

Defined In

LCCurrency

Syntax

Call currencyTotal.Add(currency1, currency2)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| currency1 | LCCurrency. The first of the two LCCurrencies to add. |
| currency2 | LCCurrency. The second of the two LCCurrencies to add. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim num1 As New LCCurrency
    Dim num2 As New LCCurrency
    Dim sum As New LCCurrency

    num1.Value = 12345.6789
    num2.Value = 12345.6789
    Call sum.Add (Num1, Num2)

    Print "The sum of the two currencies is " & sum.Text

End Sub
```

Example Output

The sum of the two currencies is 24691.3578

Compare Method for LCCurrency

This method compares two currency values and returns the relationship.

Defined In

LCCurrency

Syntax

Result = thisCurrency.Compare(baseCurrency)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| baseCurrency | LCCurrency. The base currency value to which to compare thisCurrency. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| Result | Result of the comparison: Result > 0 (positive): thisCurrency is greater than baseCurrency. Result < 0 (negative): thisCurrency is less than baseCurrency. Result = 0: thisCurrency is equal to baseCurrency |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim num1 As New LCCurrency

//For LEI syntax, see overview.

Dim num2 As New LCCurrency

num1.Value = 123.456789

num2.Value = 123

If (num1.Compare (num2) = 0) Then

Print "The first number is the same as the second."

Elseif (num1.Compare (num2) > 0) Then

Print "The first number is greater than the second."

Else

Print "The first number is less than the second."

End If

End Sub

Example Output

The first number is greater than the second.

Copy Method for LCCurrency

This method makes a copy of an LCCurrency object.

Defined In

LCCurrency

Syntax

Set newCurrency = srcCurrency.Copy

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| srcCurrency | LCCurrency. The source currency value to be copied. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| newCurrency | LCCurrency. The copy of the srcCurrency object. |

Example

Option Public

```
Uselsx "*lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim num1 As New LCCurrency
```

```
    //For LEI syntax, see overview.
```

```
    Dim num2 As LCCurrency
```

```
    num1.Value = 12345.6789
```

```
    Set num2 = num1.Copy
```

```
    Print "The copy has a value of " & num2.Text
```

```
End Sub
```

Example Output

The copy has a value of 12345.6789

Subtract Method for LCCurrency

This method subtracts one LCCurrency value from another, producing the result.

Defined In
LCCurrency

Syntax
Call currencyResult.Subtract(currency1, currency2)

Parameters

| Value | Description |
|-----------|---|
| currency1 | LCCurrency. This is the initial LCCurrency from which you want to subtract the second currency. |
| currency2 | LCCurrency. The LCCurrency to be subtracted from currency1. |

Example

```
Option Public

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim num1 As New LCCurrency //For LEI syntax, see overview.
    Dim num2 As New LCCurrency
    Dim diff As New LCCurrency

    num1.Value = 98765.4321
    num2.Value = 12345.6789

    Call diff.Subtract (Num1, Num2)

    Print "The difference of the two currencies is " & diff.Text

End Sub
```

Example Output
The difference of the two currencies is 86419.7532

Chapter 4

LCDatetime Class

This chapter provides information about the Lotus Connectors LCLdatetime class methods and properties.

Overview

The LCLdatetime class represents a specific date and time, including time zone and daylight savings time information. A datetime value may have either its date or time component unavailable, indicated by special constant values for date (LCLDTNULL_DATE) or time (LCLDTNULL_TIME) components. Flags control the behavior for the existence/absence of the date or time portion. During any datetime overflow, the maximum or minimum valid datetime value is assigned in addition to the error return.

The time portion of the datetime is precise to hundredths. Some data sources do not support this precision, or support greater precision. For these systems it may be necessary to set the field flags for all datetime datatype fields to LCFIELDF_TRUNC_PREC, to avoid a precision loss error.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "1sxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "1sxlei"
```

In addition, do the following:

- Dim LCSession as a named session (this becomes the default Log Document name)
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "*lsxlei"
```

Sub Initialize

```
Dim MyLEISession As New LCSession ("MainSession")
```

```
Dim MyNotesConnection as New LCConnection  
("MyNotesConn")
```

```
Dim MyOracleConnection as New LCConnection  
("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

Type DATETIME Format and Values

The following table lists DATETIME formats and values.

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| Type constant | LCTYPE_DATETIME |
| Description | Date and time value with time zone and Daylight Savings Time (DST) |
| Other | Precision = 0.01 second Minimum Year Value (LCMIN_DATETIME_YEAR) = 1 Maximum Year Value (LCMAX_DATETIME_YEAR) = 32767 |

LCDatetime Class Methods Summary

The following table summaries LCDatetime class methods.

| <i>Value</i> | <i>Description</i> |
|-------------------------------|---|
| Type constant | LCTYPE_DATETIME |
| Description | Date and time value with time zone and Daylight Savings Time (DST) |
| Other | Precision = 0.01 second Minimum Year Value (LCMIN_DATETIME_YEAR) = 1 Maximum Year Value (LCMAX_DATETIME_YEAR) = 32767 |
| <i>LCDatetime.SetConstant</i> | Produces a special constant Datetime commonly used for comparisons. |
| <i>LCDatetime.SetCurrent</i> | Sets a Datetime value to the current system time. |

LCDatetime Properties Summary

The following table summarizes LCDatetime properties.

| <i>Values</i> | <i>Description</i> |
|-----------------------------|---|
| <i>LCDatetime.Minute</i> | Long. Minute (0-59). |
| <i>LCDatetime.Second</i> | Long. Second (0-59). |
| <i>LCDatetime.Hundredth</i> | Long. Hundredths of second (0-99). |
| <i>LCDatetime.Day</i> | Long. Day of month (1-31). |
| <i>LCDatetime.Hour</i> | Long. Hour (0-23). |
| <i>LCDatetime.Month</i> | Long. Month (1-12). |
| <i>LCDatetime.Year</i> | Long. Integer indicating the year (1-32767). |
| <i>LCDatetime.Weekday</i> | Long. Integer indicating the day of the week (1-7, Sunday = 1). Output only. |
| <i>LCDatetime.Zone</i> | Long. Integer indicating the time zone (-12 to 12). |
| <i>LCDatetime.DST</i> | Boolean. Indicates whether Daylight Savings Time is in effect. |
| <i>LCDatetime.Ticks</i> | Long. Tick count representing hundredths of a second since midnight. |
| <i>LCDatetime.Text</i> | String representation. |
| <i>LCDatetime.Julian</i> | Long. Integer representing the Julian date by indicating the number of days since January 1, 4713 BC. |
| <i>LCDatetime.Value</i> | LotusScript Variant containing a date/time. |

New Method for LCDatetime

This is the constructor for LCDatetime. It initializes a new LCDatetime object.

Defined In
LCDatetime

Syntax
Dim *variableName* as **New** LCDatetime(*Year, Month, Day, Hour, Minute, Second, Hundredth, Zone, DST*)

Parameters
All parameters are optional.

| <i>Values</i> | <i>Description</i> |
|------------------|--|
| <i>Year</i> | As Long. Value in the range 1-32767. Default is 0. |
| <i>Month</i> | As Long. A value in the range 1-12. Default is 0. |
| <i>Day</i> | As Long. A value in the range 1-31. Default is 0. |
| <i>Hour</i> | As Long. A value in the range 0-23. Default is 0. |
| <i>Minute</i> | As Long. A value in the range 0-59. Default is 0. |
| <i>Second</i> | As Long. A value in the range 0-59. Default is 0. |
| <i>Hundredth</i> | As Long. A value in the range 0-99. Default is 0. |
| <i>Zone</i> | As Long. A value representing the time zone in the range -11 to 11. Default is GMT. |
| <i>DST</i> | As Variant. Boolean. Daylight savings time is in effect when true. Default is 0 (GMT). |

Example
`Option Public`
`Usesx "*lsxlc"`
`// DECS only, see overview in this chapter for LEI syntax.`
`Sub Initialize`
`Dim clock As New LCDatetime`

Adjust Method for LCDatetime

This method alters a datetime by specified datetime units.

Defined In
LCDatetime

Syntax
Call *changeDatetime.Adjust(Units, Amount)*

Parameters

| Values | Description |
|---------------|--|
| <i>Units</i> | Long. The unit to adjust. Units are in the order listed in the constructor: Year, Month, Day, Hour, Minute, Second, Hundredth, Zone, DST. Use the following constants: LCDTUNIT_YEAR – Year units. LCDTUNIT_MONTH – Month units. LCDTUNIT_DAY – Day units. LCDTUNIT_WEEKDAY – Weekday units. LCDTUNIT_HOUR – Hour units. LCDTUNIT_MINUTE – Minute units. LCDTUNIT_SECOND – Second units. LCDTUNIT_HUNDREDTH – Hundredth of second units. LCDTUNIT_ZONE – Time zone units – see below. |
| <i>Amount</i> | Long. The amount to adjust the datetime unit. Positive means later in time; negative means earlier. |

Time Zone Adjustment

The LCDatetime.Adjust method does not directly support time zone adjustment. To adjust a time zone, do the following:

1. Decompose the value into parts using the LCDatetime.GetPart method, for example:

```
LCDATETIME theDatetime
LCDATETIMEPARTS dtparts
theDatetime.getParts (dtparts)
```

This results in a structure with a field called lcZone.

2. Change the zone value manually, for example:
`dtparts.lcZone = dtparts.lcZone - 3`

3. Reconstruct the date and time using the LCDatetime.SetParts method, for example:

```
theDatetime.setParts (dtparts)
```

Example

```
Option Public
```

```
Uselsx "*lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
Dim clock As New LCDateTime
```

```
clock.SetCurrent
```

```
Print "The time is " & clock.Text
```

```
Call clock.Adjust (LCDTUNIT_HOUR, -100)
```

```
Print "100 hours ago, the time was " & clock.Text
```

```
End Sub
```

Example Output

```
The time is 09/08/1998 05:22:07.18 PM
```

```
100 hours ago, the time was 09/05/1998 02:37:52.82 AM
```

Clear Method for LCDatetime

This method clears a previously set Datetime value to NULL.

Defined In

LCDatetime

Syntax

Call *dateTimeObject*.Clear

Example

```
Option Public
```

```
Uselsx "*lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
Dim Clock As New LCDateTime (1999, 12, 31, 23, 59, 59, 99)
```

```
Clock.Clear
```

```
If (Clock.Text = "") Then
```

```

        Print "The time has been cleared."
    Else
        Print "The time is " & Clock.Text
    End If
End Sub

```

Example Output

The time has been cleared.

Compare Method for LCDatetime

This method compares two Datetime values and returns the relationship.

Defined In

LCDatetime

Syntax

Result = *thisDatetime.Compare(baseDatetime)*

Parameters

baseDatetime The Datetime to which to compare *thisDatetime*.

Return Value

| <i>Values</i> | <i>Description</i> |
|---------------|---|
| <i>Result</i> | Result of the comparison: Result > 0 (positive): <i>thisDatetime</i> is greater than <i>baseDatetime</i> . Result < 0 (negative): <i>thisDatetime</i> is less than <i>baseDatetime</i> . Result = 0: <i>thisDatetime</i> is equal to <i>baseDatetime</i> . |

Example

```

Option Public

Uselstx "1stxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim Clock As New LCDatetime (1999, 12, 31, 23, 59, 59, 99)

    Dim Match As New LCDatetime

    Match.SetCurrent

    If (Clock.Compare (Match) = 0) Then

        Print "The current time matches " & Clock.Text
    End If
End Sub

```

```

Elseif (Clock.Compare (Match) > 0) Then
    Print "The current time is before " & Clock.Text
Else
    Print "The current time is after " & Clock.Text
End If
End Sub

```

Example Output

The current time is before 12/31/1999 11:59:59.99 PM

Copy Method for LCDatetime

This method makes a copy of an LCDatetime object.

Defined In

LCDatetime

Syntax

Set *newDatetime* = *srcDatetime*.Copy

Parameters

SrcDatetime LCDatetime. The datetime object that you want to copy.

Return Value

newDatetime LCDatetime. The copy of the srcDatetime object.

Example

```

Option Public
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim session As New LCSession
    Dim StopTime As New LCDatetime
    Dim StartTime As LCDatetime
    StopTime.SetCurrent
    Set StartTime = StopTime.Copy
    session.Sleep (100) ' 100 milliseconds
    StopTime.SetCurrent

```



```

Print "The difference between start and stop is:"

Print StopTime.GetDiff (StartTime, LCDTUNIT_HUNDREDTH) & _
    " hundredths of a second."

End Sub

```

Example Output

```

The difference between start and stop is:
10 hundredths of a second.

```

GetDiff Method for LCDatetime

This method gets the difference between the values of two LCDatetime objects.

Defined In
LCDatetime

Syntax
difference = *lcDatetime*.**GetDiff**(*baseDatetime*, *Units*)

Parameters

| <i>Values</i> | <i>Description</i> |
|---------------------|---|
| <i>baseDatetime</i> | LCDatetime. The datetime object to which you want to compare the lcDatetime object. |
| <i>Units</i> | <p>Long. The units to compare. Units are in the order listed in the constructor: Year, Month, Day, Hour, Minute, Second, Hundredth, Zone, DST. Use the following constants:</p> <p>LCDTUNIT_YEAR – Year units.</p> <p>LCDTUNIT_MONTH – Month units.</p> <p>LCDTUNIT_DAY – Day units.</p> <p>LCDTUNIT_WEEKDAY – Weekday units.</p> <p>LCDTUNIT_HOUR – Hour units.</p> <p>LCDTUNIT_MINUTE – Minute units.</p> <p>LCDTUNIT_SECOND – Second units.</p> <p>LCDTUNIT_HUNDREDTH – Hundredth of second units.</p> <p>LCDTUNIT_ZONE – Time zone units.</p> |

Return Value
difference The difference between the two LCDatetime objects, in the units specified.

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim Boston As New LCDatetime _

(1998, 1, 25, 08, 50, 00, 00, -5, True)

Dim Singapore As New LCDatetime _

(1998, 1, 27, 1, 10, 00, 00, 8, False)

Print "A flight taking off from Boston at 8:50am"

Print "and landing in Singapore at 1:10am"

Print "will take " & Singapore.GetDiff (Boston,
LCDTUNIT_HOUR) & " hours."

End Sub

Example Output

A flight taking off from Boston at 8:50am

and landing in Singapore at 1:10am

will take 27 hours.

SetConstant Method for LCDatetime

This method produces a special constant Datetime object.

Defined In

LCDatetime

Syntax

Call *lcdatetime*.SetConstant(*datetimeConstant*)

Parameters

| Values | Description |
|-------------------------|--|
| <i>datetimeConstant</i> | One of the following Datetime Constants: LCDTCONST_MINIMUM: A datetime which is less than any other datetime (except another minimum). LCDTCONST_MAXIMUM: A datetime that is greater than any other datetime (except another maximum). LCDTCONST_WILDCARD: A datetime that matches any other datetime. The date value is LCDTNUL_DATE and the time value is LCDTNUL_TIME. |

Example

Option Public

Uselx "xlxl"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim Clock As New LCDatetime (1999, 12, 31, 23, 59, 59, 99)

Dim Match As New LCDatetime

Call Match.SetConstant (LCDTCONST_WILDCARD)

If (Clock.Compare (Match) = 0) Then

Print "The current time matches the wild card constant."

Else

Print "The current time does not match the wild card
constant."

End If

End Sub

Example Output

The current time matches the wild card constant.

SetCurrent Method for LCDatetime

This method sets the Datetime value to the current system datetime.

Defined In

LCDatetime

Syntax

Call *lcdatetimeObject*.SetCurrent

Example

Option Public

Usesx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim stopWatch As New LCDatetime

stopWatch.setcurrent

Print "The time is " & stopWatch.text

End Sub

Example Output

The time is 09/08/1999 05:22:02.85 PM

Chapter 5

LCField Class

This chapter provides information about the LCField class methods and properties.

Overview

The LCField class represents a data object containing one or more data values of a designated data type. A field may be an independent repository for data or may be a reference to another field, as in when getting a field from a fieldlist. In this case, changes to data in the field affect the contents of the data referenced by the fieldlist.

Note that in repetitive operations such as data fetch – insert loops, it is more efficient to get a reference field from a fieldlist prior to the loop, and act on the data through the reference than to locate the data field in each iteration of the loop.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session (this becomes the default Log Document name)
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
    Usesx "*lsxlei"

Sub Initialize

    Dim MyLEISession As New LCSession ("MainSession")

    Dim MyNotesConnection as New LCConnection
        ("MyNotesConn")

    Dim MyOracleConnection as New LCConnection
        ("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

LCField Class Methods Summary

The methods for the LCField class are summarized below.

Allocation

The following methods create and free Field instances.

| <i>Value</i> | <i>Description</i> |
|-----------------------|---|
| <i>New LCField</i> | (Constructor) Allocates a new Field object instance. |
| <i>LCField.Copy</i> | Creates a new Field object instance as a copy of another field. The field data is also copied. |
| <i>Delete LCField</i> | (Destructor) Frees a Field object instance allocated with the constructor or LCField.Copy. This method does not free fields created as part of a fieldlist. |
| <i>Nopyew LCField</i> | (Constructor) Allocates a new Field object instance. |
| <i>LCField.C</i> | Creates a new Field object instance as a copy of another field. The field data is also copied. |

Field Properties

The following methods support retrieval of field properties and assignment of mutable field properties.

| <i>Value</i> | <i>Description</i> |
|--------------------------------|--|
| <i>LCField.GetType</i> | Retrieves the data type for a field. |
| <i>LCField.GetValueCount</i> | Retrieves the number of data values for a field. |
| <i>LCField.GetFlags</i> | Retrieves the current field flags for a field. |
| <i>LCField.SetFlags</i> | Assigns the current field flags for a field. |
| <i>LCField.GetVirtualCode</i> | Checks if a field has this virtual code set. |
| <i>LCField.ListVirtualCode</i> | Lists the virtual code(s) for a field. |
| <i>LCField.SetVirtualCode</i> | Sets a virtual code for a field. |

Field Format

The following methods support retrieval of field format information. Setting the format for a field clears all current data values for that field.

| <i>Value</i> | <i>Description</i> |
|----------------------------------|---|
| <i>LCField.GetFormatNumber</i> | Retrieves the current format settings for a number field. |
| <i>LCField.SetFormatNumber</i> | Assigns the current format settings for a number field. |
| <i>LCField.GetFormatDatetime</i> | Retrieves the current format settings for a datetime field. |
| <i>LCField.SetFormatDatetime</i> | Assigns the current format settings for a datetime field. |
| <i>LCField.GetFormatStream</i> | Retrieves the current format settings for a stream field. |
| <i>LCField.SetFormatStream</i> | Assigns the current format settings for a stream field. |

Field Data

The following methods support access to and modification of field data and NULL indicators.

| <i>Value</i> | <i>Description</i> |
|--------------------------------|--|
| <i>LCField.IsNull</i> | Queries whether a specific field data value is NULL. |
| <i>LCField.SetNull</i> | Assigns the NULL indicator for a specific field data value. |
| <i>LCField.Get<Type></i> | Retrieves a specific field data value as a particular data type, converting the data if necessary. |
| <i>LCField.Set<Type></i> | Assigns a specific field data value from a particular data type, converting the data if necessary. |

Miscellaneous

The following methods provide compare and conversion functionality.

| <i>Value</i> | <i>Description</i> |
|------------------------|---|
| <i>LCField.Compare</i> | Compare data values between two fields. |
| <i>LCField.Convert</i> | Convert data values between two fields. |

LCField Properties

The following are the properties for the LCField class:

| <i>Value</i> | <i>Description</i> |
|----------------------------|--|
| <i>Count</i> | Long. One or greater. Data space for this many data values and NULL indicators will be allocated. The value count is automatically assigned for fields in a fieldlist based on the fieldlist record count (see Fieldlist description). Assigned at creation and Read-Only. |
| <i>Datatype</i> | Long. One of the Lotus Connector datatypes. Assigned at creation and Read-Only. |
| <i>Flags</i> | Long. Zero or more of the following field flags ORed together. |
| <i>LCFIELDF_NO_NULL</i> | Field cannot be NULL. |
| <i>LCFIELDF_TRUNC_PREC</i> | Allow precision truncation. |
| <i>LCFIELDF_TRUNC_DATA</i> | Allow data truncation. |
| <i>LCFIELDF_NO_FETCH</i> | Do not FETCH this field. |
| <i>LCFIELDF_NO_INSERT</i> | Do not INSERT this field. |
| <i>LCFIELDF_NO_UPDATE</i> | Do not UPDATE this field. |
| <i>LCFIELDF_NO_REMOVE</i> | Do not REMOVE this field. |
| <i>LCFIELDF_NO_CREATE</i> | Do not CREATE this field. |
| <i>LCFIELDF_NO_DROP</i> | Do not DROP this field. |
| <i>LCFIELDF_KEY</i> | Field is a KEY for keyed operations. |
| <i>LCFIELDF_KEY_GT</i> | Key condition is greater than. |
| <i>LCFIELDF_KEY_LT</i> | Key condition is less than. |
| <i>LCFIELDF_KEY_NE</i> | Key condition is not equal to. |
| <i>LCFIELDF_KEY_LIKE</i> | Key condition is like (native pattern match). |
| <i>IsNull</i> | Boolean. True or False. |
| <i>Text</i> | Array of string representations. |
| <i>Value</i> | Array of LotusScript datatypes. The value or values contained depend on the datatype of the field, as listed below. |
| | <i>Field Type</i> |
| | <i>LCTYPE_CURRENCY</i> |
| | <i>LCTYPE_DATETIME</i> |
| | <i>LCTYPE_INT</i> |
| | <i>LCTYPE_FLOAT</i> |
| | <i>LCTYPE_NUMERIC</i> |
| | <i>LCTYPE_TEXT</i> |
| | <i>LCTYPE_BINARY</i> |

Field Format

The format of a field is specific to its general class of type: number (int, float, currency, numeric), datetime, or stream (text and binary). For all format values, zero indicates that the specified information is not used for this field. The information for each datatype is given below.

Note Number and Datetime formats do not affect the actual data, but are used for data creation only. Stream format does affect the actual data.

Number (INT, FLOAT, CURRENCY, NUMERIC)

Flags: zero or more of the following constants. Multiple constants can be ORed together:

| <i>Value</i> | <i>Description</i> |
|---------------------------|---|
| <i>LCNUMBERF_UNSIGNED</i> | Source type is unsigned. |
| <i>LCNUMBERF_NUMERIC</i> | Source type is NUMERIC (unnecessary with LCTYPE_NUMERIC). |
| <i>LCNUMBERF_DECIMAL</i> | Source type is DECIMAL. |
| <i>LCNUMBERF_PACKED</i> | Source numeric/decimal data is packed. |
| <i>LCNUMBERF_BIT</i> | Source type is a bit. |

Size: size, in bytes, of this number value. Zero indicates to default to the size of the type for this field.

Precision: digits of precision for this value. Zero indicates not used.

Scale: numeric scale. A true scale of zero uses the constant LCSCALE_ZERO. Zero indicates not used.

Datetime (DATETIME)

Flags: zero or more of the following constants. Multiple constants can be ORed together:

| <i>Value</i> | <i>Description</i> |
|----------------------------|---------------------------|
| <i>LCDATETIMEF_NO_DATE</i> | Source type is time only. |
| <i>LCDATETIMEF_NO_TIME</i> | Source type is date only. |

Size: size, in bytes, of this datetime value. Zero indicates to default to the size of the type for this field.

Stream (TEXT, BINARY)

Flags: zero or more of the stream flags LCSTREAMF_xxx. Multiple flags can be ORed together. See the Stream class description.

MaxLength: maximum length, in bytes, of this stream value. Zero indicates no maximum length. See the Stream class description.

StreamFormat: default stream format for this stream value. Zero indicates no default stream format. Use one of the LCSTREAMFMT_xxx constants.

Field Virtual Codes – Advanced Usage

Virtual codes allow specific fields to be interpreted differently for Connectors which support virtual fields. For example, a Connector could support a virtual field named “RecordId,” which holds a special internal record indicator for that Connector. This Connector would interpret this field differently than normal data, while other Connectors would consider this field a standard data field. While the virtual code indicates special handling, the field name determines the type of handling on a Connector-specific basis.

To achieve this functionality, the virtual code for a field is set to match either a Connector code or a Connection code. A Connector code is a code which is the same for all connections to a particular Connector in a Session. The Connector code can be obtained by retrieving the property `LCTOKEN_CONNECTOR_CODE` with `LCConnection.GetProperty` or as the code returned by `LCSession.ListConnector` or `LCSession.LookupConnector`. Using a Connector code as a virtual code causes all connections to that Connector to interpret the field as a virtual field. A connection code is a code which is different for all connections. The Connect code can be obtained by retrieving the property `LCTOKEN_CONNECTION_CODE` with `LCConnection.GetProperty`. Using a Connection code as a virtual code causes only that specific Connection to interpret the field as a virtual field. The Connector code can also be determined by taking the Connection code and zeroing the low two bytes (OR with `LCMASK_CONNECTOR_CODE`). Note that these codes are dynamically assigned and must be obtained for each execution of a script.

Processing Attachments Stored in RTF Fields

Notes is primarily a document management system, not a relational database management system (RDBMS). Because of this, Notes performs special handling for attachments stored in Rich Text fields. It stores the attachment data separately from the Rich Text field data. Currently you can only transfer attachments from Notes to Notes; Rich Text fields with attachments do not store correctly in the target RDBMSs such as DB2 and Oracle.

To work around this constraint, you can use the Notes Connector option **Extract File Attachments** to save attachments to files on disk. You can then use LotusScript to read the files, store their contents in `LCFields`, and save the `LCFields` to binary fields in a RDBMS. This level of scripting requires LotusScript expertise.

New Method for LCField

This is the constructor method for LCField. It initializes an LCField object.

Defined In
LCField

Syntax
Dim variablename as New LCField(type, count)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| <i>type</i> | Long. Data type of the field object, one of the following constants: LCTYPE_CURRENCY LCTYPE_DATETIME LCTYPE_INT LCTYPE_FLOAT LCTYPE_NUMERIC LCTYPE_TEXT LCTYPE_BINARY |
| <i>count</i> | Long. Optional. Number of data values to be allocated for this field. The default is 1. |

Example

```
Option Public
Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim session As New LCSession //For LEI syntax, see overview.
    Dim field As New LCField (LCTYPE_INT)
```

ClearVirtualCode Method for LCField

This method clears the specified virtual code for the field.

Defined In
LCField

Syntax
Call field.ClearVirtualCode(*code*)

Parameters

code The code to clear. Zero (0) clears all codes.

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
    Dim field As New LCField (LCTYPE_INT)
    Dim Code As Long
    Dim text As String
    Call session.ListConnector (LCLIST_FIRST, , Code)
    Call field.SetVirtualCode (Code)
    While session.ListConnector (LCLIST_NEXT, , Code)
        Call field.SetVirtualCode (Code)
    Wend

    ' use the value to clear an individual connector's virtual
code
    ' use 'zero' to clear all connectors' virtual codes
    Call field.ClearVirtualcode (0)
    Print "All of the virtual codes have been cleared."
End Sub
```

Example Output

All of the virtual codes have been cleared.

Compare Method for LCField

This method compares data values between two fields and returns the relationship.

Defined In

LCField

Syntax

Result = *thisField.Compare*(*thisIndex*, *baseField*, *baseIndex*, *valueCount*, *compareFlags*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>thisIndex</i> | Long. Index identifying the first value to be compared from <i>thisField</i> . |
| <i>baseField</i> | LCField. Base field for comparison. |
| <i>baseIndex</i> | Long. Index identifying the first base value in <i>baseField</i> . |
| <i>valueCount</i> | Long. Number of field values to be compared. Starting at the indicated index in each field, ValueCount consecutive fields from each fieldlist will be compared until all are compared or until there is a mismatch. |
| <i>compareFlags</i> | Long. Zero or more of the following values, ORed together: LCCOMPAREF_NO_ZONE – Do not consider time zone or daylight savings time information in datetime comparisons. LCCOMPAREF_SECOND – Do not consider fractions of seconds in datetime comparisons. LCCOMPAREF_FLOAT_PREC – Compare floating point values to only ten digits of precision. In addition, the following composite flag is supplied: LCCOMPAREF_LOW_PREC – Composite of all low-precision LCCOMPAREF flags. |

Return Value

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| <i>Result</i> | Result of the comparison: Result > 0 (positive): The first mismatch that the function encountered in <i>thisField</i> is greater than the corresponding value in <i>baseField</i> . Result < 0 (negative): The first mismatch that the function encountered in <i>thisField</i> is less than the corresponding value in <i>baseField</i> . Result = 0: Each compared value in <i>thisField</i> is equal to the corresponding value in <i>baseField</i> . |

Example

```
Option Public

Uselxx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field1 As New LCField (LCTYPE_INT, 3)

    Dim field2 As New LCField (LCTYPE_INT, 2)
```

```

Call field1.SetInt (1, 100)
Call field1.SetInt (2, 300)
Call field1.SetInt (3, 400)
Call field2.SetInt (1, 300)
Call field2.SetInt (2, 400)
If (field1.Compare(2, field2, 1, 2, 0) <> 0) Then
    Print "the 2nd and 3rd values of field1 " & _
    "do not match the 1st and 2nd values of field2."
Else
    Print "the 2nd and 3rd values of field1 " & _
    "match the 1st and 2nd values of field2."
End If
End Sub

```

Example Output
the 2nd and 3rd values of field1 match the 1st and 2nd values
of field2.

Convert Method for LCField

This method converts data values to the data type of a destination field.

Defined In
LCField

Syntax

Call *thisField.Convert*(thisIndex, srcField, srcIndex, valueCount)

Parameters

| <i>Value</i> | <i>Description</i> |
|-------------------|--|
| <i>thisIndex</i> | Long. Index identifying the first target data value in <i>thisField</i> . |
| <i>srcField</i> | LCField. Source field for the data values. Values will be converted to the data type of <i>thisField</i> . |
| <i>srcIndex</i> | Long. Index identifying the first source data value to be converted in <i>SrcField</i> . |
| <i>valueCount</i> | Long. Number of field values to be converted. If the end of either field is encountered before converting this number of data values, then the function stops at that point. |
| <i>thisIndex</i> | Long. Index identifying the first target data value in <i>thisField</i> . |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim f_field As New LCField (LCTYPE_FLOAT)

    Dim c_field As New LCField (LCTYPE_CURRENCY)

    f_field.value = 12345.123456789

    Call c_field.Convert (1, f_field, 1, 1)

    Print "The float field is " & f_field.text(0) & "and it was"

    Print "converted to a currency field as " & c_field.text(0)

End Sub
```

Example Output

```
The float field is 12345.123456789 and it was
converted to a currency field as 12345.1235
```

Copy Method for LCField

This method creates a field with the settings and values of a source field. The new field contains the same number of data values of the same data type as the source field. The new values are copies of, rather than references to, the original data.

Defined In

LCField

Syntax

Set *newField* = *origField*.Copy

Parameters

origField LCField. The field object that you want to copy.

Return Value

newField LCField. The copy of the original field.

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
```

```

Sub Initialize

    Dim field As New LCField (LCTYPE_FLOAT)

    Dim c_field As LCField

    field.value = 12345.123456789

    Set c_field = field.Copy

    Print "The float field is " & field.text(0)

    Print "and its copy is " & c_field.text(0)

End Sub

```

Example Output

The float field is 12345.123456789

and its copy is 12345.123456789

GetCurrency Method for LCField

This method retrieves a Currency data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newCurrency* = *lcField*.GetCurrency(*index*)

Parameters

| | |
|--------------|---|
| <i>index</i> | Long. Index identifying the field data value to be retrieved. |
|--------------|---|

Return Value

| | |
|--------------------|---------------------------------------|
| <i>newCurrency</i> | The value of the retrieved data type. |
|--------------------|---------------------------------------|

Example

```

Option Public

Usesx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim fld As New LCField (LCTYPE_TEXT)

    fld.Text = "1234.56789"

```



```

Dim vCurr As LCCurrency

Set vCurr = Fld.GetCurrency (1)

Print "The Currency representation of the field is
      " & vCurr.Text

End Sub

```

Example Output

The Currency representation of the field is 1234.5678

GetDatetime Method for LCField

This method retrieves a Datetime data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set newDatetime = lcField.GetDatetime(index)

Parameters

index Long. Index identifying the field data value to be retrieved.

Return Value

newDatetime The value of the retrieved data type.

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Fld As New LCField (LCTYPE_TEXT)

    Fld.Text = "1234.56789"

    Dim vDate As LCDateTime

    Fld.Text = "10/12/1998"

    Set vDate = Fld.GetDatetime (1)

    Print "The Datetime representation of the field is
          " & vDate.Text

End Sub

```

Example Output

The Datetime representation of the field is 10/12/1998

GetFieldlist Method for LCField

This method retrieves a fieldlist from a field. The resulting fieldlist is a reference to the original inside the field. If the actual field data type is different, an LCFAIL_INVALID_CONVERT error will occur.

Defined In

LCField

Syntax

Set *newFieldList* = *lcField*.GetFieldList(*index*)

Parameters

index Long. Index identifying the field data value to be retrieved.

Return Value

newFieldList The value of the retrieved data type

Example

Option Public

Option Explicit

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim Record As New LCFieldlist

Dim SubRecord As New LCFieldlist

Dim field As LCField

REM start building FieldList

Set field = Record.Append ("group", LCTYPE_INT)

field.Value = 4200

REM Build SubFieldList

Set field = SubRecord.Append ("category", LCTYPE_TEXT)

field.Value = "potato"

Set field = SubRecord.Append ("description", LCTYPE_TEXT)

field.Value = "russet"

```

Set field = SubRecord.Append ("sku", LCTYPE_INT)
field.Value = 4207
REM return to building the FieldList
Set field = Record.Append ("item", LCTYPE_FIELDLIST)
REM now assign the SubRecord to the Record
Call field.SetFieldList (1, SubRecord)
Delete SubRecord
Set SubRecord = field.getFieldList (1)
Print "The first field of the field's fieldlist is
      " & SubRecord.Names(0)
End Sub

```

Example Output

The first field of the field's fieldlist is category.

GetFloat Method for LCField

This method retrieves a LotusScript double data type from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newFloat* = *lcField*.**GetFloat**(*index*)

Parameters

index Long. Index identifying the field data value to be retrieved.

Return Value

newFloat The value of the retrieved data type.

Example

```

Option Public

Uselsx "xlsxl"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim fld As New LCField (LCTYPE_TEXT)

```

```

Fld.Text = "1234.56789"

Dim vFloat As Double

vFloat = Fld.GetFloat (1)

Print "The Float representation of the field is " & vFloat

End Sub

```

Example Output

The Float representation of the field is 1234.56789

GetFormatDatetime Method for LCField

This method retrieves the format of a Datetime type field. It is only valid for fields of type Datetime.

Use the LCField.SetFormatDatetime method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField.GetFormatDatetime(datetimeFlags, size)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>datetimeFlags</i> | Long. Optional. Datetime flags of the field. Refer to the Field Format Datetime section for a description of the flags. |
| <i>size</i> | Long. Optional. Size in bytes of the datetime field. Zero indicates the size of an a Lotus Connector Datetime (8bytes). |

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_DATETIME, 1)

    Dim flags As Long

    Dim size As Long

    Call field.SetFormatDatetime (LCDATETIMEF_NO_TIME, 0)

```

```

    Call field.GetFormatDateTime (flags, size)

    Print "The datetime format flag setting is " & Hex(flags)
        & "h"

End Sub

```

Example Output

Error! Cannot open file.

GetFormatNumber Method for LCField

This method retrieves the format of a number type field. It is only valid for fields of type int, currency, float or numeric.

Use the LCField.SetFormatNumber method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField*.GetFormatNumber(*numberFlags*, *size*, *precision*, *scale*)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|---|
| <i>numberFlags</i> | Long. Optional. Number flags of the number field. Refer to the Field Number Format section for a description of the flags. The default is Nothing. |
| <i>size</i> | Long. Optional. Size in bytes of the number field. Zero indicates the size of the corresponding number object (LONG, DOUBLE, LCCURRENCY, or LCNUMERIC). The default is Nothing. |
| <i>precision</i> | Long. Optional. Precision of the number field. Zero indicates not used for this field. The default is Nothing. |
| <i>scale</i> | Long. Optional. Scale of the number field. Zero indicates not used for this field. Since zero is also a valid scale value, a constant LCSCALE_ZERO is defined which can be used to indicate a zero scale. The default is Nothing. |
| <i>numberFlags</i> | Long. Optional. Number flags of the number field. Refer to the Field Number Format section for a description of the flags. The default is Nothing. |

Example

```
Option Public

Uselsx "xlsxl"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_NUMERIC)

    Dim flags As Long

    Call field.SetFormatNumber (LCNUMBERF_UNSIGNED, , 10, 4)

    Call field.GetFormatNumber (flags)

    Print "The number flag setting is " & Hex(flags) & "h"

End Sub
```

Example Output

The number flag setting is 1h

GetFormatStream Method for LCField

This method retrieves the format of a stream type field. It is only valid for fields of type text or binary.

Use the LCField.SetFormatStream method to assign stream format information. Refer to the Field Format section for a description of format values.

Defined In

LCField

Syntax

Call *thisField*.GetFormatStream(*streamFlags*, *maxLength*, *streamFormat*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>streamFlags</i> | Long. Optional. Stream flags for the field. Refer to the Stream Class section for a description of the flags. The default is Nothing. |
| <i>maxLength</i> | Long. Optional. Maximum length of the stream field. A value of zero indicates no maximum length. The default is Nothing. |
| <i>streamFormat</i> | Long. Optional. Stream format of the stream field. A value of zero indicates no specified stream format. The default is Nothing. |
| <i>streamFlags</i> | Long. Optional. Stream flags for the field. Refer to the Stream Class section for a description of the flags. The default is Nothing. |
| <i>maxLength</i> | Long. Optional. Maximum length of the stream field. A value of zero indicates no maximum length. The default is Nothing. |

Example

```
Option Public

Uselstx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_BINARY)

    Dim fmt As Long

    Dim flags As Long

    Call field.SetFormatStream (LCSTREAMF_NO_CASE, 256,
        LCSTREAMFMT_BIG5)

    Call field.GetFormatStream (flags, , fmt)

    Print "The stream format and flag settings are:"

    Print "format=" & fmt & " flags=" & Hex(flags) & "h"

End Sub
```

Example Output

```
The stream format and flag settings are:

format=26 flags=10h
```

GetInt Method for LCField

This method retrieves an integer from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newInt* = *thisField*.*GetInt(index)*

Parameters

index Long. Index identifying the field data value to be retrieved.

Return Value

newInt The value of the retrieved data type.

Example

```
Option Public
```

```

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Fld As New LCField (LCTYPE_TEXT)

    Fld.Text = "1234.56789"

    Dim vInt As Long

    vInt = Fld.GetInt (1)

    Print "The Int representation of the field is " & vInt

End Sub

```

Example Output

The Int representation of the field is 1234

GetNumeric Method for LCField

This method retrieves a numeric value from a value in a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newNumeric* = *thisField*.GetNumeric(*index*)

Parameters

index Long. Index identifying the field data value to be retrieved.

Return Value

newNumeric The value of the retrieved data type.

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Fld As New LCField (LCTYPE_TEXT)

    Fld.Text = "1234.56789"

    Dim vNumr As LCNumeric

```



```

Set vNumr = Fld.GetNumeric (1)

Print "The Numeric representation of the field is
      " & vNumr.Text

End Sub

```

Example Output

The Numeric representation of the field is 1234.56789

GetStream Method for LCField

This method retrieves a stream value from a field. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Set *newStream* = *thisField*.**GetStream** (*index*, *streamFormat*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|--|
| <i>index</i> | Long. Index identifying the field data value to be retrieved. |
| <i>streamFormat</i> | Long. Stream format to obtain the data in. Any valid stream format is allowed. In addition, a value of zero will either use the stream format of the field (if thisField is TEXT or BINARY), or convert to native text format LCSTREAMFMT_NATIVE (if thisField is not a local type). |

Return Value

newStream The value of the retrieved data type.

Example

```

Option Public

Uselstx "1sxlst"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Fld As New LCField (LCTYPE_TEXT)

    Fld.Text = "1234.56789"

    Dim vStrm As LCStream

    Set vStrm = Fld.GetStream (1, LCSTREAMFMT_ASCII)

```

```

    Print "The Stream representation of the field is
        " & vStrm.Text

End Sub

```

Example Output

The Stream representation of the field is 1234.56789

IsNull Method for LCField

This method returns true or false depending on whether the specified field data is NULL.

Defined in
LCField

Syntax

flag = *thisfield.IsNull(index)*

Parameters

index Long. Index identifying the field.

Return Value

flag Boolean value, either TRUE or FALSE.

Example

```

Option Public

Uselsx "1sxlxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_TEXT)

    Dim clock As New LCDatetime

    Print "Before setting the value is the field NULL?
        " & field.IsNull(1)

    clock.SetCurrent

    Call field.SetDatetime (1, clock)

    Print "After setting the value is the field NULL?
        " & field.IsNull(1)

End Sub

```

Example Output

Before setting the value is the field NULL? True

After setting the value is the field is NULL? False

LookupVirtualCode Method for LCField

This method checks if a specific VirtualCode has been set for a field.

Defined In

LCField

Syntax

Flag = *field*.LookupVirtualCode(*virtualCode*)

Parameters

virtualCode The virtual code to look for on the list of codes for this field.

Return Value

Flag TRUE if this virtual code is set for the field; FALSE otherwise.

Example

```
Option Public

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
    Dim field As New LCField (LCTYPE_INT)
    Dim Code As Long
    Dim text As String

    Call session.ListConnector (LCLIST_FIRST, , Code)
    Call field.SetVirtualCode (Code)
    While session.ListConnector (LCLIST_NEXT, , Code)
        Call field.SetVirtualCode (Code)
    Wend

    If (field.LookupVirtualCode (&H10000)) Then
        Print "The field has virtual code 10000h set."
    Else
        Print "The field does not have virtual code 10000h set."
    End If
End Sub
```

Example Output

The field has virtual code 10000h set.

SetCurrency Method for LCField

This method assigns the currency value to the specified data index of the field. If the field data type is different, conversion will be attempted.

Defined In

LCField

Syntax

Call *thisField.SetCurrency(index, srcCurrency)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| <i>index</i> | Long. Index indentifying the value that is to be assigned. |
| <i>srcCurrency</i> | LCCurrency. Value to be assigned to the field data value. |

Example

```
Option Public

Uselsx "*1sxlx"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_TEXT)

    Dim number As New LCCurrency

    number.value = 1234567890.1234

    Call field.SetCurrency (1, number)

    Print "The field's value is " & field.text(0)

End Sub
```

Example Output

The field's value is 1234567890.1234

SetDatetime Method for LCField

This method assigns the datetime value to the specified index of the field. If the field data type is different, conversion will be attempted.

Defined In

LCField

Syntax

Call *thisField.SetDateTime(index, srcDateTime)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| <i>index</i> | Long. Index indentifying the value that is to be assigned. |
| <i>srcDateTime</i> | LCDateTime. Value to be assigned to the field data value. |

Example

Option Public

```
Usesx "*lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim field As New LCField (LCTYPE_TEXT)
```

```
    Dim clock As New LCDateTime
```

```
    clock.SetCurrent
```

```
    Call field.SetDatetime (1, clock)
```

```
    Print "The field's value is " & field.text(0)
```

```
End Sub
```

Example Output

```
The field's value is 09/08/1998 05:22:30.86 PM
```

SetFieldlist Method for LCField

This method assigns the fieldlist value to the specified index of the field. If the field data type is different, an LCFAIL_INVALID CONVERT error will occur.

Defined In

LCField

Syntax

Call *thisField*.SetFieldlist(index, srcFieldlist)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|--|
| <i>index</i> | Long. Index indentifying the value that is to be assigned. |
| <i>srcFieldlist</i> | LCFieldlist. Value to be assigned to the field data value. |

Example

```
Option Public

Option Explicit

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim Record As New LCFieldList

    Dim SubRecord As New LCFieldList

    Dim field As LCField

    REM start building FieldList

    Set field = Record.Append ("group", LCTYPE_INT)

    field.Value = 4200

    REM Build SubFieldList

    Set field = SubRecord.Append ("category", LCTYPE_TEXT)

    field.Value = "potato"

    Set field = SubRecord.Append ("description", LCTYPE_TEXT)

    field.Value = "russet"

    Set field = SubRecord.Append ("sku", LCTYPE_INT)

    field.Value = 4207

    REM return to building the FieldList

    Set field = Record.Append ("item", LCTYPE_FIELDLIST)

    REM now assign the SubRecord to the Record

    Call field.SetFieldList (1, SubRecord)

    REM Take a look at the Record while debugging the
        LotusScript

    Print "The sub fieldlist has successfully been appended to
        parent fieldlist as another field."

End Sub
```

Example Output

```
The sub fieldlist has successfully been appended to parent
    fieldlist as another field.
```

SetFloat Method for LCField

This method assigns a value to a field of type float. If the field data type is different, conversion will be attempted.

Defined In
LCField

Syntax
Call *thisField.SetFloat(index, srcFloat)*

Parameters

| Value | Description |
|---------------------|--|
| <i>index</i> | Long. Index identifying the value that is to be assigned. |
| <i>srcFieldlist</i> | LCFieldlist. Value to be assigned to the field data value. |

Example

```
Option Public
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim field As New LCField (LCTYPE_TEXT)
    Dim number As Double
    number = Pi
    Call field.SetFloat (1, number)
    Print "The field's value is " & field.text(0)
End Sub
```

Example Output
The field's value is 3.14159265358979

SetFormatDatetime Method for LCField

This method assigns the current format setting for a datetime field. Calling this method clears all values for this field.

Defined In
LCField

Syntax
Call *thisField.SetFormatDatetime(datetimeFlags, size)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|---|
| <i>srcDateTime</i> | LCDateTime. Value to be assigned to the field data value. The <i>datetimeFlags</i> do not affect the data. As with the <i>size</i> value, it is only relevant when interacting with the external system. The flags will have no effect on the conversion of a field's value to or from a stream or text. |
| <i>size</i> | Long. Optional. Size in bytes assigned to the datetime field. Zero indicates the size of an Lotus Connector Datetime (8 bytes). The default is 0. The <i>size</i> value is used for metadata creation by data systems which support datetime datatypes of different sizes. For systems which support only a single datetime datatype, the value of <i>size</i> is ignored. The default value for <i>size</i> is zero. For example, Sybase supports both a 4-byte datetime as well as an 8-byte version. When building up a fieldlist that will be used to create a Sybase table (see LCConnection.Create for an example), use the SetFormatDatetime method to indicate which size datetime datatype should be used. |

Example

Option Public

```
Useslx "lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim field As New LCField (LCTYPE_DATETIME, 1)
```

```
    Dim flags As Long
```

```
    Dim size As Long
```

```
    Call field.SetFormatDatetime (LCDATETIMEF_NO_TIME, 0)
```

```
    Call field.GetFormatDateTime (flags, size)
```

```
    Print "The datetime format flag setting is " & Hex(flags)  
        & "h"
```

```
End Sub
```

Example Output

The datetime format flag setting is 2h

SetFormatNumber Method for LCField

This method assigns the current format setting for a number field. Calling this method clears all values for this field.

Defined In
LCField

Syntax
Call *thisField.SetFormatNumber (numberFlags, size, precision, scale)*

Parameters

| Value | Description |
|--------------------|---|
| <i>numberFlags</i> | Long. Optional. Number flags assigned to the number field. Refer to the Field Number Format section for a description of the flags. The default is 0. |
| <i>size</i> | Long. Optional. Size in bytes assigned to the number field. Zero indicates the size of the corresponding number object (LONG, DOUBLE, LCCURRENCY, or LCNUMERIC). The default is 0. |
| <i>precision</i> | Long. Optional. Precision assigned to the number field. Zero indicates not used for this field. The default is LCMAX_NUMERIC_PREC. |
| <i>scale</i> | Long. Optional. Scale assigned to the number field. Zero indicates not used for this field. Since zero is also a valid scale value, a constant LCSCALE_ZERO is defined which can be used to indicate a zero scale. The default is LCMAX_NUMERIC_PREC / 2. |

Example

```
Option Public
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim field As New LCField (LCTYPE_NUMERIC)
    Dim flags As Long
    Call field.SetFormatNumber (LCNUMBERF_UNSIGNED, , 10, 4)
    Call field.GetFormatNumber (flags)
    Print "The number flag setting is " & Hex(flags) & "h"
End Sub
```

Example Output
The number flag setting is 1h

SetFormatStream Method for LCField

This method assigns the current format setting for a stream field. Calling this method clears all values for this field.

Defined In
LCField

Syntax
Call *thisField.SetFormatStream(streamFlags, maxLength, streamFormat)*

Parameters

| Value | Description |
|---------------------|--|
| <i>streamFlags</i> | Long. Optional Stream flags assigned to the field. Refer to the Stream Class section for a description of the flags. The default is 0. |
| <i>maxLength</i> | Long. Optional Maximum length assigned to the stream field. A value of zero indicates no maximum length. The default is 0. |
| <i>streamFormat</i> | Long. Optional Stream format assigned to the stream field. A value of zero indicates no specified stream format. The default is LCSTREAMFMT_UNICODE. |

Example

```
Option Public

Useslx "slxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_BINARY)

    Dim fmt As Long

    Dim flags As Long

    Call field.SetFormatStream (LCSTREAMF_NO_CASE, 256,
        LCSTREAMFMT_BIG5)

    Call field.GetFormatStream (flags, , fmt)

    Print "The stream format and flag settings are:
        format=" & fmt & " flags=" & Hex(flags) & "h"

End Sub
```

Example Output
The stream format and flag settings are: format=26 flags=10h

SetInt Method for LCField

This method assigns a value to a field of type long. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In
LCField

Syntax
Call *thisField.SetInt* (index, srcInt)

Parameters

| Value | Description |
|---------------|---|
| <i>index</i> | Long. Index identifying the value that is to be assigned. |
| <i>srcInt</i> | Long. Value to be assigned to the field data value. |

Example

```
Option Public
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim field As New LCField (LCTYPE_TEXT)
    Dim number As Long
    number = Pi
    Call field.SetInt (1, number)
    Print "The field's value is " & field.text(0)
End Sub
```

Example Output
The field's value is 3

SetNumeric Method for LCField

This method assigns a value to a field of type LCNumeric. If the requested data type is different from the field data type, conversion is automatically performed.

Defined In
LCField

Syntax

Call *thisField.SetNumeric*(index, srcNumeric)

Parameters

| <i>Value</i> | <i>Description</i> |
|-------------------|---|
| <i>index</i> | Long. Index identifying the value that is to be assigned. |
| <i>srcNumeric</i> | LCNumeric. Value to be assigned to the field data value. |

Example

Option Public

Use!sx "1sxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim field As New LCField (LCTYPE_TEXT)

Dim number As New LCNumeric

number.value = 1234567890.12345678

Call field.SetNumeric (1, number)

Print "The field's value is " & field.text(0)

End Sub

Example Output

The field's value is 1234567890.12346

SetStream Method for LCField

This method assigns a value to a field of type LCStream (text or binary). If the requested data type is different from the field data type, conversion is automatically performed.

Defined In

LCField

Syntax

Call *thisfield.SetStream*(index, srcStream)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|---|
| <i>index</i> | Long. Index identifying the value that is to be assigned. |
| <i>srcStream</i> | LCStream. Value to be assigned to the field data value. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim field As New LCField (LCTYPE_TEXT)

    Dim msg As New LCStream

    msg.Text = "Hello World"

    Call field.SetStream (1, msg)

    Print "The field's value is " & field.text(0)

End Sub
```

Example Output

The field's value is Hello World

SetVirtualCode Method for LCField

This method adds a virtual code to the list of virtual codes for a field.

Defined In

LCField

Syntax

Call *thisField*.SetVirtualCode(*virtualCode*)

Parameters

virtualCode Long. VirtualCode value to assign to *thisField*.

Example

```
Option Public

Uselsx "*lsxlc"
    // decs only, see overview in this chapter for lei syntax.

Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.

    Dim field As New LCField (LCTYPE_INT)

    Dim Code As Long

    Dim text As String

    Call session.ListConnector (LCLIST_FIRST, , Code)
```

```

Call field.SetVirtualCode (Code)
text = Hex(Code) & "h"
While session.ListConnector (LCLIST_NEXT, , Code)
    Call field.SetVirtualCode (Code)
    text = text & ", " & Hex(Code) & "h"
Wend
Print "The field virtual codes (one per connector) are " &
    text
End Sub

```

Example Output

The field virtual codes (one per connector) are 10000h,
20000h, 30000h, 40000h, 50000h

Chapter 6

LCFieldlist Class

This chapter provides information about the LCFieldlist class methods and properties.

Overview

The LCFieldlist class represents metadata (the description of data from a data source) for a record and may reference data, in the form of LCFields, for one or more record values. A fieldlist is a list of fields and field names with a represented order. Fields within a fieldlist can be added, modified, retrieved, or listed in multiple ways. The fields, field names, and order are separate entities and only have relation within an individual fieldlist.

Note that many situations exist where the names or order must be changed to accommodate different connections, but the data needs to remain constant. In these cases, more than one fieldlist may be created, each referenced by the same fields using the Map, MapName, or Merge methods.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session (this becomes the default Log Document name)
- LCCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "*lsxlei"
```

Sub Initialize

```
Dim MyLEISession As New LCSession ("MainSession")  
Dim MyNotesConnection as New LCConnection ("MyNotesConn")  
Dim MyOracleConnection as New LCConnection  
("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

Accessing Field Data

Field data within an LCFieldlist object may be accessed as a property by name. The datatype of the property depends on the datatype of the field. The property is an array of LotusScript datatypes, as shown in the table below. For example, if a fieldlist has a field named OrderID, its first value may be obtained with the script:

```
Identification = LCFieldlist.OrderID(0)
```

Likewise the field may be set with the syntax:

```
LCFieldlist.OrderID = "Bart-001"
```

The LCFieldlist object may contain one or more values for each field contained within, depending on the parameters to the constructor. If the object is constructed for more than a single value, subsequent values may be accessed through the field names by changing the subscript. Multiple values may be set by assigning an array.

| <i>Field Type</i> | <i>Value Return Type</i> |
|-------------------|--------------------------|
| LCTYPE_CURRENCY | Currency |
| LCTYPE_DATETIME | Variant (Date/Time) |
| LCTYPE_INT | Long |
| LCTYPE_FLOAT | Double |
| LCTYPE_NUMERIC | Double |
| LCTYPE_TEXT | String |
| LCTYPE_BINARY | String |

Fieldlist Merging and Mapping

Fieldlist merging with `LCFieldlistMerge` and `LCFieldlistMergeVirtual` support field mapping. To perform mapping, two fieldlists are provided, one as the name source and the other as the field/data source. Depending on merge options, a mapping is produced between the names of the name source and the fields of the data source, and a third fieldlist is produced which references parts of the original fieldlists. When using virtual fields and `LCFieldlistMergeVirtual`, fields in the data fieldlist which match the supplied virtual code are excluded from mapping and are added to a separate new virtual fieldlist.

Mapping and Merging

When a fieldlist is to be used for more than one specific action, there is often a need to have it treated differently by each of those actions. For example, fields in a fieldlist map need to be reordered, renamed, or removed, but only in the context of one particular operation. For such situations, four methods are provided to perform this mapping: `Map`, `MapName`, `Merge`, and `MergeVirtual`. `MergeVirtual` is simply a variation of `Merge` which handles virtual codes as well. These methods produce a new fieldlist from an existing fieldlist, but with the same fields referenced from both fieldlists, allowing two different 'maps' into a set of fields. If these two fieldlists are sent to different Connector methods, then the Connectors will see different sets of fields in a different order, but referring to the same data and common fields.

The three methods `Map`, `MapName`, and `Merge` have various levels of difficulty and control available to the caller; which one is needed depends on the specific situation. In general, the simplest method which supports your requirements should be used. These methods are listed below, starting with the simplest.

`Map` is applicable when the names in the fieldlist are correct, but fields either need to be reordered or removed. It accepts a fieldlist and a text list of names, and the new fieldlist contains the fields from the original fieldlist reordered in the order they are listed in the text list. In addition, any fields not included in the text list are excluded from the new fieldlist.

`MapName` adds renaming of fields to `Map`, but is otherwise identical. By adding a second text list of names, it will rename each field in the first text list with the corresponding entry in the second text list.

`Merge` accepts two fieldlists, one which provides the list of fields, and one which provides the list of names. In addition, it accepts a set of flags which further control the new fieldlist produced. These flags control the type of mapping, whether to allow fields to be excluded from either fieldlist, and how to interact with specific field flags. `Merge` should only be used when the

functionality of Map and MapName is insufficient for your needs, as Merge is much more complex to use.

LCFieldlist Class Methods Summary

Creation

The following methods create and free Fieldlist instances. The Fieldlist Merging methods also create new fieldlists.

| <i>Value</i> | <i>Description</i> |
|----------------------------|---|
| <i>New LCFieldlist</i> | (Constructor) Creates a new Fieldlist object instance. |
| <i>LCFieldlist.Copy</i> | Creates a new Fieldlist object instance as a copy of another fieldlist. The field data is also copied. |
| <i>LCFieldlist.CopyRef</i> | Creates a new Fieldlist object instance as a partial copy of another fieldlist. The contained fields are not copied, but are rather referenced. |
| <i>Delete LCFieldlist</i> | (Destructor) Frees a Fieldlist object instance created with the constructor, <i>LCFieldlist.Copy</i> , <i>LCFieldlist.CopyRef</i> , <i>LCFieldlist.Merge</i> , or <i>LCFieldlist.MergeVirtual</i> . |

Fieldlist Retrieval

The following methods support retrieval of fieldlist contents.

| <i>Value</i> | <i>Description</i> |
|-----------------------------|---|
| <i>LCFieldlist.GetField</i> | Retrieves a particular field from a fieldlist by field index. |
| <i>LCFieldlist.GetName</i> | Retrieves a copy of a field name from a fieldlist by field index. |
| <i>LCFieldlist.Lookup</i> | Locates a field in a fieldlist by field name. |

Fieldlist Modification

The following methods support modification of fieldlist contents.

| <i>Value</i> | <i>Description</i> |
|-----------------------------|---|
| <i>LCFieldlist.GetField</i> | Retrieves a particular field from a fieldlist by field index. |
| <i>LCFieldlist.GetName</i> | Retrieves a copy of a field name from a fieldlist by field index. |
| <i>LCFieldlist.Lookup</i> | Locates a field in a fieldlist by field name |
| <i>LCFieldlist.GetField</i> | Retrieves a particular field from a fieldlist by field index. |
| <i>LCFieldlist.GetName</i> | Retrieves a copy of a field name from a fieldlist by field index. |

Fieldlist Merging

The following methods support fieldlist merging.

| <i>Value</i> | <i>Description</i> |
|---------------------------------|--|
| <i>LCFieldlist.Merge</i> | Merges a name fieldlist and data fieldlist to produce a mapped fieldlist. |
| <i>LCFieldlist.MergeVirtual</i> | Merges a name fieldlist and data fieldlist to produce a mapped fieldlist. Fields with matching virtual codes are added to a new virtual fieldlist. |

Fieldlist Mapping

The following methods support fieldlist mapping.

| <i>Value</i> | <i>Description</i> |
|----------------------------|--|
| <i>LCFieldlist.Map</i> | Maps fields with the same names but changes the positions and can exclude some fields. |
| <i>LCFieldlist.MapName</i> | Maps fields with different names. |

LCFieldlist Properties Summary

The following is a list of fieldlist properties.

| <i>Value</i> | <i>Description</i> |
|----------------------------|--|
| <i>LCFieldlist.Map</i> | Maps fields with the same names but changes the positions and can exclude some fields. |
| <i>LCFieldlist.MapName</i> | Maps fields with different names. |
| <i>LCFieldlist.Map</i> | Maps fields with the same names but changes the positions and can exclude some fields. |
| <i>LCFieldlist.MapName</i> | Maps fields with different names. |
| <i>LCFieldlist.Map</i> | Maps fields with the same names but changes the positions and can exclude some fields. |

All other properties are dynamic. Field data may be accessed by referencing the field name.

New Method for LCFieldlist

This is the constructor for LCFieldlist.

Defined In

LCFieldlist

Syntax

Dim *variableName* as **New** LCFieldlist(*recordCount*, *fieldFlags*)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|---|
| <i>recordCount</i> | Long. Optional. The number of records in the fieldlist. The default is 1. All fields added to the fieldlist will have this number of records. |
| <i>fieldFlags</i> | Long. Optional. The default field flags for the fieldlist (zero or more LCFIELD_XXX flags ORed together). These are set as the initial field flags for each field added to the fieldlist. The default is 1. |

Example

Option Public

```
Uselsx "lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
Dim src As New LCConnection ("db2")
```

```
//For LEI syntax, see overview.
```

Append Method for LCFieldlist

This method appends a field to an existing fieldlist.

Defined In

LCFieldlist

Syntax

Set *field* = *fieldlist.Append*(*fieldName*, *dataType*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|---|
| <i>fieldName</i> | String. The name for the field. |
| <i>dataType</i> | Long. The constant for the datatype. One of the following: LCTYPE_INT LCTYPE_FLOAT LCTYPE_CURRENCY LCTYPE_NUMERIC LCTYPE_DATETIME LCTYPE_TEXT LCTYPE_BINARY LCTYPE_FIELDLIST LCTYPE_CONNECTION |

Example

Option Public

Option Explicit

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

REM this example copies a DB2 table

Dim src As New LCConnection ("db2")

//For LEI syntax, see overview.

Dim fldLstRecord As New LCFieldList

Dim fld As LCField

REM build the table definition

Call fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)

Call fldLstRecord.Append ("CONTACTNAME", LCTYPE_TEXT)

Call fldLstRecord.Append ("COMPANYNAME", LCTYPE_TEXT)

REM set properties to connect to both data sources

src.Database = "Gold"

src.Userid = "JDoe"

src.Password = "xyzzzy"

src.Metadata = "customer"

REM now connect

src.Connect

REM create it based on the metadata property already set
above

On Error LCFAIL_DUPLICATE Goto tableexists

Call src.Create (LCOBJECT_METADATA, fldLstRecord)

Print "The '" & src.Metadata &
"'" table did not exist so it was created."

End

tableexists:

Print "The '" & src.Metadata & "'" table exists."

End

End Sub

Example Output

The 'customer' table exists.

Copy Method for LCFieldlist

This method creates a duplicate copy of an LCFieldlist and all its data.

Defined In
LCFieldlist

Syntax
Set fldListRecordNew = fldListRecord.Copy

Parameters
fldListRecord LCFieldlist. The fieldlist that you want to copy.

Return Value
fldListRecordNew LCFieldlist. The copy of the *fldListRecord* object.

Example

Option Public

```
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim olist As New LCFieldlist
    Dim nlist As LCFieldlist
    Dim ofield As LCField

    Set ofield = olist.Append ("FirstName", LCTYPE_TEXT)
    ofield.Text = "Chi Len"

    Set nlist = olist.Copy
    Set nfield = nlist.GetField (1)
    Call olist.SetName (1, "FullName")
    ofield.Text = "Cheiko"

    Print "The copy contains:"

    Print "field named " & nlist.Names(0) & " whose value is "
        & nfield.Text(0)

End Sub
```

Example Output

The copy contains:

field named FirstName whose value is Chi Len

CopyField Method for LCFieldlist

This method copies an existing field within a fieldlist at a specified position.

Defined In
LCFieldlist

Syntax
Set *newField* = *fieldList.CopyField(index, srcField, name)*

Parameters

| <i>Value</i> | <i>Description</i> |
|-----------------|---|
| <i>index</i> | Long. Position of the field to copy. |
| <i>srcField</i> | LCField. The source field to copy. |
| <i>name</i> | String. The name to give the copy of the field. |

Return Value

name String. The name of the copy of the field.

Example

```
Option Public
Option Explicit
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim fldLstRecord As New LCFieldList
    Dim fld As New LCField (LCTYPE_TEXT)
    Dim ref As LCField
    Dim text As String
    ' There are a number of ways to build a fieldlist
    ' Append will add a field to a list given a type
    Set ref = fldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
    Set ref = fldLstRecord.Append ("COMPANYID", LCTYPE_INT)
    ' Insert is like Append but the position within
    ' the fieldlist must be specified
    Set ref = fldLstRecord.Insert (1, "COMPANYADDRESS",
    LCTYPE_TEXT)
    ' CopyField will add a field to a list
```

```

' using an existing field as a template
fld.Flags = LCFIELD_KEY
Set ref = FldLstRecord.CopyField (1, fld, "CONTACTNAME")
' IncludeField will add an existing field to a list,
' making it part of the list. In this case, 'fld'
' becomes a reference into the fieldlist
fld.Flags = 0
Call FldLstRecord.IncludeField (3, fld, "COMPANYCITY")
text = ""
Forall fieldname In FldLstRecord.Names
    If Text = "" Then text = fieldname Else text = text + ", " & fieldname
End Forall
Print "The field list looks like: " & text
End Sub

```

Example Output

The field list looks like: CONTACTNAME, COMPANYADDRESS, COMPANYCITY, ACCOUNTMANAGER, COMPANYID

CopyRef Method for LCFieldlist

This method creates a new fieldlist object instance as a partial copy of another fieldlist. The fields in the original fieldlist are not copied, but are referenced.

This method creates a new fieldlist with separate name space but with the same fields and data space as an existing fieldlist. The field names are copied, but the new fieldlist references the original fields. In this situation, any changes to the data of one fieldlist is actually a change to the data for all fieldlists referencing the same data space.

To copy a fieldlist's metadata and data, use `LCFieldlistCopy`.

Defined In

LCFieldlist

Syntax

Set *fldListRecordNew* = *fldListRecord*.CopyRef

Parameters

fldListRecord LCFieldlist. The fieldlist that you want to make a reference copy of.

Return Values

FldListRecordNew LCFieldlist. The reference copy of the fieldlist.

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim olist As New LCFieldlist

Dim nlist As LCFieldlist

Dim ofield As LCField

Set ofield = olist.Append ("FirsName", LCTYPE_TEXT)

ofield.Text = "Chi Len"

Set nlist = olist.CopyRef

Set nfield = nlist.GetField (1)

Call olist.SetName (1, "FullName")

ofield.Text = "Cheiko"

Print "The copy contains:"

Print "field named " & nlist.Names(0) & " whose value is "
`& nfield.Text(0)

End Sub

Example Output

The copy contains:

field named FirsName whose value is Cheiko

GetField Method for LCFieldlist

This method gets a field reference from a fieldlist result set and places its value into a variable.

Defined In
LCFieldlist

Syntax
Set field = fldLstRecord.GetField(index)

Parameters
index Long. Position of the field to be returned.

Return Values
field LCField. The field at the index position in the fieldlist.

Example

```
Option Public

Uselstx "lslstlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim connect As New LCConnection ("db2")
    //For LEI syntax, see overview.

    Dim conFldLst As New LCFieldList

    Dim field As LCField

    ' this section assigns the appropriate properties to connect
    ' to DB2

    connect.Database = "Gold"
    connect.Userid = "JDoe"
    connect.Password = "xyzzzy"
    connect.Metadata = "customer"

    ' connect to DB2
    connect.Connect

    ' now perform the catalog action - in this case for fields
    If (connect.Catalog (LCOBJECT_FIELD, conFldLst) = 0) Then
        Print "No tables were found."
    Else
```

```

Set field = conFldLst.GetField(1)

Print "The columns in the '" & connect.Metadata & "'
      table include:"

While (connect.Fetch (conFldLst) > 0)
    Print "      " & field.text(0)
Wend

End If

End Sub

```

Example Output

The columns in the 'CUSTOMER' table include:

```

ACCOUNTMANAGER
CONTACTNAME
COMPANYNAME
COMPANYADDRESS
COMPANYCITY
COMPANYSTATE
COMPANYPHONE

```

GetName Method for LCFieldlist

This method obtains a copy of the name of a field within a fieldlist.

Defined In
LCFieldlist

Syntax

fieldName = fieldListRecord.**GetName**(*index*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------------|---|
| <i>fieldListRecord</i> | String. The fieldlist from which to retrieve the fieldname. |
| <i>index</i> | Long. The position of the field in the fieldlist. |

Return Values

fieldName String. The name of the field.

Example

```
Option Public
```

```

Uselsx "*lsxlc"
  // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

  Dim connect As New LCConnection ("db2")
  //For LEI syntax, see overview.

  Dim FldLst As New LCFieldlist

  Dim field As LCField

  Dim i As Long

  ' this section assigns the appropriate properties to connect
  'to DB2

  connect.Database = "Gold"
  connect.Userid = "JDoe"
  connect.Password = "xyzyzzy"
  connect.Metadata = "customer"
  connect.Connect

  ' now perform the catalog action - in this case for fields
  If (connect.Select (Nothing, 1, FldLst) = 0) Then
    Print "The customer table was not found."
  Else
    Print "There are " & FldLst.FieldCount & _
      " columns in the '" & connect.Metadata & "' table."
    Print "They are:"
    For i = 1 To FldLst.FieldCount
      Print Tab(4); FldLst.GetName (i)
    Next
  End If
End Sub

```

Example Output

There are 7 columns in the 'customer' table.

They are:

```

ACCOUNTMANAGER
CONTACTNAME
COMPANYNAME

```

COMPANYADDRESS
COMPANYCITY
COMPANYSTATE
COMPANYPHONE

IncludeField Method for LCFieldlist

This method includes an existing individual field into a fieldlist.

Defined In
LCFieldlist

Syntax

Call *fldLstRecord.IncludeField(index, field, fieldName)*

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>index</i> | Long. Position of the field within fldLstRecord. |
| <i>field</i> | LCField. The new field. |
| <i>fieldName</i> | String. The name of the new field. |

Example

```
Option Public
Option Explicit

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim fldLstRecord As New LCFieldList
    Dim fld As New LCField (LCTYPE_TEXT)
    Dim ref As LCField
    Dim text As String

    ' There are a number of ways to build a fieldlist
    ' Append will add a field to a list given a type
    Set ref = FldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)
    Set ref = FldLstRecord.Append ("COMPANYID", LCTYPE_INT)
    ' Insert is like Append but the position
```

```

' within the fieldlist must be specified
Set ref = FldLstRecord.Insert (1, "COMPANYADDRESS", LCTYPE_TEXT)
' CopyField will add a field to a list
' using an existing field as a template
fld.Flags = LCFIELD_KEY
Set ref = FldLstRecord.CopyField (1, fld, "CONTACTNAME")
' IncludeField will add an existing field to a list,
' making it part of the list. in this case, 'fld'
' becomes a reference into the fieldlist
fld.Flags = 0
Call FldLstRecord.IncludeField (3, fld, "COMPANYCITY")
text = ""
Forall fieldname In FldLstRecord.Names
    If Text = "" Then text = fieldname Else text = text + ", "
    + fieldname
End Forall
Print "The field list looks like: " & text
End Sub

```

Example Output

The field list looks like: CONTACTNAME, COMPANYADDRESS,
COMPANYCITY, ACCOUNTMANAGER, COMPANYID

Insert Method for LCFieldlist

This method inserts a new field into an existing fieldlist at a specified index position.

Defined In
LCFieldlist

Syntax

Set *fieldNew* = *fldLstRecord.Insert* (*index*, *fieldName*, *dataType*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>index</i> | Long. Index number of the field before which the new field is inserted. An index number equal to the number of fields currently in the fieldlist (plus one) is equivalent to using LCFieldlist.Append. |
| <i>fieldName</i> | String. Name of the new field. |
| <i>dataType</i> | Long. Data type of the new field. One of the following: LCTYPE_INT LCTYPE_FLOAT LCTYPE_CURRENCY LCTYPE_NUMERIC LCTYPE_DATETIME LCTYPE_TEXT LCTYPE_BINARY LCTYPE_FIELDLIST LCTYPE_CONNECTION |

Example

Option Public

Option Explicit

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim fldLstRecord As New LCFieldList

Dim fld As New LCField (LCTYPE_TEXT)

Dim ref As LCField

Dim text As String

' There are a number of ways to build a fieldlist

' Append will add a field to a list given a type

Set ref = FldLstRecord.Append ("ACCOUNTMANAGER", LCTYPE_INT)

Set ref = FldLstRecord.Append ("COMPANYID", LCTYPE_INT)

' Insert is like Append but the position

' within the fieldlist must be specified

Set ref = FldLstRecord.Insert (1, "COMPANYADDRESS", LCTYPE_TEXT)

' CopyField will add a field to a list

```

' using an existing field as a template
fld.Flags = LCFIELD_KEY
Set ref = FldLstRecord.CopyField (1, fld, "CONTACTNAME")
' IncludeField will add an existing field to a list,
' making it part of the list. in this case, 'fld'
' becomes a reference into the fieldlist
fld.Flags = 0
Call FldLstRecord.IncludeField (3, fld, "COMPANYCITY")
text = ""
Forall fieldname In FldLstRecord.Names
    If Text = "" Then text = fieldname Else text = text + ", "
    + fieldname
End Forall
Print "The field list is: " & text
End Sub

```

Example Output

The field list is: CONTACTNAME, COMPANYADDRESS, COMPANYCITY,
ACCOUNTMANAGER, COMPANYID

List Method for LCFieldlist

This method iterates through fields in a fieldlist, optionally returning information.

Defined In
LCFieldlist

Syntax

Call *fldLstRecod.List*(*position, destField, index, dataType, flags, fieldName*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>position</i> | Long. Constant indicating whether to return the first or next Connector Property. The options are: LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this function for this Connection). |
| <i>destField</i> | LCField. Optional. The field in the fieldlist. |
| <i>index</i> | Long. Optional. Index of destField in the fieldlist. This may not advance consecutively for fieldlists produced with LCFieldlist.Merge or LCFieldlist.MergeVirtual. |
| <i>dataType</i> | Long. Optional. Data type of destField. |
| <i>flags</i> | Long. Optional. Field flags of destField. Refer to the Field Class section for a description of the flags. |
| <i>fieldName</i> | String. Optional. Copy of the name of destField. |

Example

Option Public

```
Uselsx "*lsxlc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim connect As New LCConnection ("db2")
```

```
    //For LEI syntax, see overview.
```

```
    Dim FldLst As New LCFieldlist
```

```
    Dim pos As Long
```

```
    Dim dtype As Long
```

```
    Dim flags As Long
```

```
    Dim fieldname As String
```

```
    REM this section assigns the appropriate properties to  
connect to DB2
```

```
    connect.Database = "Gold"
```

```
    connect.Userid = "JDoe"
```

```
    connect.Password = "xyzzzy"
```

```
    connect.Metadata = "customer"
```

```
    REM connect to DB2
```

```
    connect.Connect
```

```
    REM now perform the catalog action - in this case for fields
```

```

If (connect.Select (Nothing, 1, FldLst) = 0) Then
    Print "The customer table was not found."
Else
    Print "The table description is:"
    pos = LCLIST_FIRST
    While (FldLst.List (pos, , , dtype, flags, fieldname) =
        True)
        Print "    " + fieldname + " is type #" + _
            Cstr(dtype) + " with flags " + Hex(flags)
        pos = LCLIST_NEXT
    Wend
End If
End Sub

```

Example Output

The table description is:

```

ACCOUNTMANAGER is type #1 with flags 2
CONTACTNAME is type #6 with flags 2
COMPANYNAME is type #6 with flags 2
COMPANYADDRESS is type #6 with flags 2
COMPANYCITY is type #6 with flags 2
COMPANYSTATE is type #6 with flags 2
COMPANYPHONE is type #6 with flags 2

```

Lookup Method for LCFieldlist

This method locates a field from a fieldlist based on field name.

Defined In
LCFieldlist

Syntax

Set *field* = *fldListRecord*.**Lookup** (*fieldName*, *index*)

Parameters

fieldName String. Name of the field to look up.

Return Value

index Long. Optional. Position of the field.

Example

Option Public

Uselsx "*lsxlc"

 // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

 Dim connect As New LCConnection ("db2")

 //For LEI syntax, see overview.

 Dim FldLst As New LCFieldlist

 Dim field As LCField

 Dim pos As Long

 REM this section assigns the appropriate properties to
connect to DB2

 connect.Database = "Gold"

 connect.Userid = "JDoe"

 connect.Password = "xyzyzy"

 connect.Metadata = "customer"

 REM connect to DB2

 connect.Connect

 REM now perform the catalog action - in this case for fields

 If (connect.Select (Nothing, 1, FldLst) = 0) Then

 Print "The customer table was not found."

 Else

 Set field = FldLst.Lookup ("contactname", pos)

 If Not (field Is Nothing) Then

 Print "Found 'ContactName' in the fieldlist"

 Print "at position " & pos

 Else

 Print "Did not find 'ContactName' in the fieldlist."

 End If

 End If

End Sub

Example Output

```
Found 'ContactName' in the fieldlist  
at position 2
```

Map Method for LCFieldlist

This method remaps fields in a fieldlist.

Use this method for field mapping operations when the fieldnames are the same in source and target, but you need to change the order of the fields or you wish to exclude some fields.

Defined In

LCFieldlist

Syntax

Call *fldListRecord.Map*(*baseFieldList*, *nameList*)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>baseFieldList</i> | LCFieldlist. The fieldlist to map. |
| <i>nameList</i> | String. Comma-delimited names of fields from baseFieldlist in the new, remapped order. |

Example

Option Public

Useslx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim session As New LCSession

Dim srcCon As New LCConnection ("db2")

//For LEI syntax, see overview.

Dim fldLst As New LCFieldList

Dim fetchLst As New LCFieldList

Dim count As Long

Dim cname As LCField

Dim ccity As LCField

Dim cstate As LCField

```

' set the appropriate properties to connect to the data
'sources

srcCon.Database = "Gold"
srcCon.Userid = "JDoe"
srcCon.Password = "xyzzzy"
srcCon.Metadata = "customer"
srcCon.Connect

' perform a select and get the records with
' the fields wanted, in our fldLstRecord object
If (srcCon.Select (Nothing, 1, fldLst) <> 0) Then
    ' now map the fields of interest, in the order needed
    Call fetchLst.Map (fldLst, "ContactName, CompanyCity,
        CompanyState")

    Set cname = fetchLst.GetField (1)
    Set ccity = fetchLst.GetField (2)
    Set cstate = fetchLst.GetField (3)

    REM fetch a record from the result set
    srcCon.MapByName = True

    Print "Fetching ContactName, CompanyCity, and
        CompanyState fields"

    While (srcCon.Fetch (fetchLst, 1, 1) > 0)
        count = count + 1

        Print Cstr(count); Tab(3); cname.Text(0); Tab(28); _
            ccity.Text(0); Tab(42); cstate.Text(0)

    Wend

End If

End Sub

```

Example Output

Fetching ContactName, CompanyCity, and CompanyState fields

| | | | |
|---|-----------------------|-------------|------|
| 1 | Peter Pan | Never Never | Land |
| 2 | R. U. Happy | Beagle | WI |
| 3 | Issac Bernard Mathews | New York | NY |

MapName Method for LCFieldlist

This method maps fields with different names. Use this method for field mapping operations when the fieldnames are different in source and target. This allows you to change the order of fields and to exclude fields, as well.

Defined In
LCFieldlist

Syntax
Call *fldListRecod.MapName*(*baseFieldList*, *nameList*, *mapList*)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>baseFieldList</i> | LCFieldlist. The fieldlist to map. |
| <i>nameList</i> | String. The names of the original fields. |
| <i>mapList</i> | String. The new names and mapping order for the fields. |

Example

```
Option Public

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim session As New LCSession

    Dim srcCon As New LCConnection ("db2")
        //For LEI syntax, see overview.

    Dim destCon As New LCConnection ("notes")

    Dim fetchLst As New LCFieldList

    Dim insertLst As New LCFieldList

    Dim count As Long

    REM set the appropriate properties to connect to the data
        sources

    srcCon.Database = "Gold"

    srcCon.Userid = "JDoe"

    srcCon.Password = "xyzzz"

    srcCon.Metadata = "customer"

    destCon.Server = "Rainbow"

    destCon.Database = "Gold"
```

```

destCon.Metadata = "customer"

REM connect to the two data sources

srcCon.Connect

destCon.Connect

srcCon.FieldNames = "ContactName, CompanyCity, CompanyState"
If (srcCon.Select (Nothing, 1, fetchLst) <> 0) Then
    ' map the result set from the SELECT with the desired names
    Call insertLst.MapName (fetchLst, _
        "ContactName, CompanyCity, CompanyState", _
        "Name, City, State")
    ' set the property MapbyName on the target.
    ' this is necessary if the fields in the form are
    ' (or might be) in differnt order from the mapping
    destCon.MapByName = True
    While (srcCon.Fetch (fetchLst, 1, 1) > 0)
        count = count + destCon.Insert (insertLst, 1, 1)
    Wend
    Print "Transferred " & count & " records from DB2, to
        Notes"
    Print "Field mapping from (DB2 column name to Notes Field
        name):"
    Print "    ContactName    -->    Name"
    Print "    CompanyCity    -->    City"
    Print "    CompanyState    -->    State"
End If
End Sub

```

Example Output

Transferred 3 records from DB2, to Notes

Field mapping from (DB2 column name to Notes Field name):

```

ContactName    -->    Name
CompanyCity    -->    City
CompanyState    -->    State

```

Merge Method for LCFieldlist

This method merges two fieldlists, creating a new third mapping fieldlist. Use this method for field mapping operations when you either have a name fieldlist already built and/or you need to use fieldlist flags.

Defined In
LCFieldlist

Syntax
Call fldListRecord.**Merge**(nameFieldList, dataFieldList, mergeFlags)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>nameFieldlist</i> | LCFieldlist. Source fieldlist containing the field names for the new fieldlist. |
| <i>dataFieldlist</i> | LCFieldlist. Source fieldlist containing the referenced data for the new fieldlist. |
| <i>mergeFlags</i> | Long. By default, both fieldlists must have the same number of fields, and those fields are mapped by position (first to first, second to second, and so on). You can use mergeFlags to alter this default behavior, with zero or more of the following values, ORED together: LCMERGEF_MAP_NAME Match source and destination fields by field name instead of by position. LCMERGEF_DATA_LOSS Ignore fields in DataFieldlist that have no corresponding field in NameFieldlist. LCMERGEF_NAME_LOSS Ignore fields in NameFieldlist that have no corresponding field in DataFieldlist. LCMERGEF_FETCH Ignore fields with the LCFIELDF_NO_FETCH flag set. LCMERGEF_INSERT Ignore fields with the LCFIELDF_NO_INSERT flag set. LCMERGEF_UPDATE Ignore fields with the LCFIELDF_NO_UPDATE flag set. LCMERGEF_REMOVE Ignore fields with the LCFIELDF_NO_REMOVE flag set. LCMERGEF_CREATE Ignore fields with the LCFIELDF_NO_CREATE flag set. LCMERGEF_DROP Ignore fields with the LCFIELDF_NO_DROP flag set. LCMERGEF_KEY Include fields with the LCFIELDF_KEY flag set. |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim session As New LCSession

Dim srcCon As New LCConnection ("db2")

//For LEI syntax, see overview.

Dim destCon As New LCConnection ("notes")

Dim fldLst As New LCFieldlist

Dim fetchLst As New LCFieldlist

Dim insertLst As New LCFieldlist

Dim dataLst As New LCFieldlist

Dim nameLst As New LCFieldlist

Dim count As Long

REM set the appropriate properties to connect to the data
sources

srcCon.Database = "Gold"

srcCon.Userid = "JDoe"

srcCon.Password = "xyzzzy"

srcCon.Metadata = "customer"

destCon.Server = "Rainbow"

destCon.Database = "Gold"

destCon.Metadata = "customer"

REM connect to the two data sources

srcCon.Connect

destCon.Connect

REM we can perform a select and get the records with the
fields we want

REM in our fldLstRecord object

If (srcCon.Select (Nothing, 1, fldLst) <> 0) Then

REM first we identify the data fields to fetch and order
them

```

Call dataLst.Append ("CONTACTNAME", LCTYPE_TEXT)
Call dataLst.Append ("COMPANYCITY", LCTYPE_TEXT)
Call dataLst.Append ("COMPANYSTATE", LCTYPE_TEXT)
Call fetchLst.Merge (dataLst, fldLst, LCMERGEF_MAP_NAME Or
    LCMERGEF_DATA_LOSS)

REM now we need to do a merge of the fields being fetched
    with the

REM names of the fields being stored
Call nameLst.Append ("Name", LCTYPE_TEXT)
Call nameLst.Append ("City", LCTYPE_TEXT)
Call nameLst.Append ("State", LCTYPE_TEXT)
Call insertLst.Merge (nameLst, fetchLst, 0)

    REM set the property Map by Name on both data sources
srcCon.MapByName = True
destCon.MapByName = True

    REM fetch a record from the result set
While (srcCon.Fetch (fetchLst, 1, 1) > 0)

    REM now insert the record into the target and fetch the
        next,

    REM looping until all records have been inserted
    count = count + destCon.Insert (insertLst, 1, 1)
Wend

Print "Transferred " & count & " records from DB2, to
    Notes"

Print "Field mapping from (DB2 column name to Notes Field
    name):"

Print "    ContactName    -->    Name"
Print "    CompanyCity    -->    City"
Print "    CompanyState    -->    State"

End If
End Sub

```

Example Output

Transferred 3 records from DB2, to Notes

Field mapping from (DB2 column name to Notes Field name):

```
ContactName    -->  Name
CompanyCity    -->  City
CompanyState   -->  State
```

MergeVirtual Method for LCFieldlist

This method merges two fieldlists, creating new mapping and virtual fieldlists.

Note This method is provided for backward compatibility. We recommend that you use either the Map or MapName methods for merging fieldlists and creating new mappings.

Defined In
LCFieldlist

Syntax

Call *fldList.MergeVirtual(nameFieldList, dataFieldList, MergeFlags, virtualCode, virtualFieldList)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>nameFieldlist</i> | LCFieldlist. Source fieldlist containing the field list of names for the new fieldlist. |
| <i>dataFieldlist</i> | LCFieldlist. Source fieldlist containing the referenced data for the new fieldlist. |
| <i>mergeFlags</i> | Long. By default, both fieldlists must have the same number of fields, and those fields are mapped by position (first to first, second to second, and so on). Use MergeFlags to alter this default behavior. Zero or more of the following values, OR-ed together: LCMERGEF_MAP_NAME – Match source and destination fields by field name instead of by position. (See Comment.) LCMERGEF_DATA_LOSS – Ignore fields in DataFieldlist that have no corresponding field in NameFieldlist. LCMERGEF_NAME_LOSS – Ignore fields in NameFieldlist that have no corresponding field in DataFieldlist. LCMERGEF_FETCH – Ignore fields with the LCFIELDF_NO_FETCH flag set. LCMERGEF_INSERT – Ignore fields with the LCFIELDF_NO_INSERT flag set. |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| | LCMERGEF_UPDATE – Ignore fields with the LCFIELDF_NO_UPDATE flag set. |
| | LCMERGEF_REMOVE – Ignore fields with the LCFIELDF_NO_REMOVE flag set. |
| | LCMERGEF_CREATE – Ignore fields with the LCFIELDF_NO_CREATE flag set. |
| | LCMERGEF_DROP – Ignore fields with the LCFIELDF_NO_DROP flag set. |
| | LCMERGEF_KEY – Include fields with the LCFIELDF_KEY flag set. |
| <i>virtualCode</i> | Long. Connect or Connector code to separate fields with a matching virtual code into virtualFieldlist. |
| <i>virtualFieldlist</i> | LCFieldlist. New fieldlist, containing fields with virtual codes matching virtualCode. |

Remove Method for LCFieldlist

This method removes an existing field from a fieldlist.

Defined In
LCFieldlist

Syntax
Call *fldLstRecord.Remove* (*index*)

Parameters
index Long. The index position of the field to be removed from the fieldlist.

Example

```
Option Public

Uselsx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim connect As New LCConnection ("db2")
    //For LEI syntax, see overview.

    Dim fldLst As New LCFieldlist

    Dim field As LCField

    Dim text As String

    REM this section assigns the appropriate properties to
    connect to DB2
```

```

connect.Database = "Gold"
connect.Userid = "JDoe"
connect.Password = "xyzzzy"
connect.Metadata = "customer"
REM connect to DB2
connect.Connect

REM now perform the catalog action - in this case for fields
If (connect.Select (Nothing, 1, FldLst) = 0) Then
    Print "The customer table was not found."
Else
    REM fetch the field names
    Call FldLst.Remove (1)
    text = ""
    Forall fieldname In FldLst.Names
        If (text = "") Then text = fieldname Else text = text +
            ", " + fieldname
    End Forall
    Print "The new list of columns in the '" & connect.Metadata
        & "' table are: " & text
End If
End Sub

```

Example Output

The new list of columns in the 'customer' table are:
 CONTACTNAME, COMPANYNAME, COMPANYADDRESS, COMPANYCITY,
 COMPANYSATE, COMPANYPHONE

Replace Method for LCFieldlist

This method replaces a field within a fieldlist.

Defined In
LCFieldlist

Syntax

Call *fldLstRecord.Replace* (*index*, *fieldName*, *dataType*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|---|
| <i>index</i> | Long. The index position of the field to replace. |
| <i>fieldName</i> | String. The name of the new field. |
| <i>dataType</i> | Long. The datatype to assign to the new field. |

Example

Option Public

UseSlsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim connect As New LCConnection ("db2")
//For LEI syntax, see overview.

Dim fldLst As New LCFieldlist

Dim field As LCField

Dim text As String

REM this section assigns the appropriate properties to
connect to DB2

connect.Database = "Gold"

connect.Userid = "JDoe"

connect.Password = "xyzzzy"

connect.Metadata = "customer"

REM connect to DB2

connect.Connect

REM now perform the catalog action - in this case for fields

If (connect.Select (Nothing, 1, fldLst) = 0) Then

Print "The customer table was not found."

```

Else

    REM fetch the field names

    Call FldLst.Replace (2, "pinky", LCTYPE_TEXT)

    text = ""

    Forall fieldname In FldLst.Names

        If (text = "") Then text = fieldname Else text = text +
            ", " + fieldname

    End Forall

    Print "The new list of columns in the '" & connect.Metadata
        & "' table are: " & text

End If

End Sub

```

Example Output

The new list of columns in the 'customer' table are:
ACCOUNTMANAGER, pinky, COMPANYNAME, COMPANYADDRESS,
COMPANYCITY, COMPANYSTATE, COMPANYPHONE

SetName Method for LCFieldlist

This method changes the name of a field within a fieldlist.

Defined In
LCFieldlist

Syntax

Call *fldLstRecord.SetName(index, fieldName)*

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|---|
| <i>index</i> | Long. The index position of the field. |
| <i>fieldName</i> | String. The new name to give to this field. |

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

```

```

Dim connect As New LCConnection ("db2")
  //For LEI syntax, see overview.

Dim fldLst As New LCFieldlist

Dim field As LCField

Dim text As String

REM this section assigns the appropriate properties to
connect to DB2

connect.Database = "Gold"
connect.Userid = "JDoe"
connect.Password = "xyzzzy"
connect.Metadata = "customer"

REM connect to DB2
connect.Connect

REM now perform the catalog action - in this case for fields
If (connect.Select (Nothing, 1, fldLst) = 0) Then
  Print "The customer table was not found."
Else
  REM fetch the field names
  Call fldLst.SetName (1, "pinky")
  text = ""

  Forall fieldname In fldLst.Names
    If (text = "") Then text = fieldname Else text = text +
      ", " + fieldname
  End Forall

  Print "The new list of columns in the '" & connect.Metadata
    & "' table are: " & text

End If
End Sub

```

Example Output

The new list of columns in the 'customer' table are: pinky,
 CONTACTNAME, COMPANYNAME, COMPANYADDRESS, COMPANYCITY,
 COMPANYSTATE, COMPANYPHONE

Chapter 7

LCNumeric Class

This chapter provides information about the Lotus Connectors LCNumeric class methods and properties.

Overview

The LCNumeric class represents a numeric value containing a precision, scale, and variable number of digits. Numeric values have a specific precision and scale, and can accommodate very high-precision numbers. By default, a new numeric submitted to any other numeric method is initialized to a precision of 88 and a scale of 44. Note that during any numeric overflow, the maximum or minimum valid numeric value is assigned in addition to the error occurring.

For a numeric to be valid, it must have valid values for precision (number of digits) and scale (number of decimal places). The precision and scale of a numeric may only be set when it is first created. If precision and scale are not valid for a numeric submitted as parameters when creating a numeric, then an error is generated. If the invalid numeric has been zeroed, then it will be automatically initialized to a numeric with the maximum precision and a scale of precision/2 (44).

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "*lsxlei"
```

Sub Initialize

```
Dim MyLEISession As New LCSession ("MainSession")
```

```
Dim MyNotesConnection as New LCConnection ("MyNotesConn")
```

```
Dim MyOracleConnection as New LCConnection  
("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

Type NUMERIC Format and Values

The following is a list of formats and values for the NUMERIC type.

NUMERIC Formats and Values

Precision (LCMAX_NUMERIC_PREC) = 88

Minimum Scale (LCMIN_NUMERIC_SCALE) = -127

Maximum Scale (LCMAX_NUMERIC_SCALE) = 127

Minimum Positive Value (LCMIN_NUMERIC_VALUE) = 1.0e-127

Maximum Positive Value (LCMAX_NUMERIC_VALUE) = 9.99... e126

LCNumeric Class Properties Summary

The following table lists the LCNumeric class properties.

| <i>Value</i> | <i>Description</i> |
|----------------------------|---|
| <i>LCNumeric.Precision</i> | Long [Read-Only]. Precision and Scale are set when the LCNumeric object is first constructed. See the New method for LCNumeric. |
| <i>LCNumeric.Scale</i> | Long [Read-Only]. Precision and Scale are set when the LCNumeric object is first constructed. See the New method for LCNumeric. |
| <i>LCNumeric.Text</i> | String representation. |
| <i>LCNumeric.Value</i> | Double – value conversion between Lotus Connectors and LotusScript double. |

LCNumeric Class Methods Summary

The following table lists the LCNumeric class methods.

| <i>Value</i> | <i>Description</i> |
|---------------------------|---|
| <i>LCNumeric.Add</i> | Adds two numeric values, producing the sum. |
| <i>LCNumeric.Compare</i> | Compares two numeric values returning a value indicating the relationship between them. |
| <i>LCNumeric.Copy</i> | Makes a copy of a numeric value. |
| <i>LCNumeric.Subtract</i> | Subtracts one numeric value from another, producing the result. |

New Method for LCNumeric

This is the constructor method for the new LCNumeric class object.

Defined In

LCNumeric

Syntax

Dim variableName As New LCNumeric(long1, long2)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| <i>long1</i> | Long. Optional. Precision for this object, from 0 - 88. Default is LCMAX_NUMERIC_PREC (88). |
| <i>long2</i> | Long. Optional. Scale for this object, from -127 to +127. Default is LCMAX_NUMERIC_PREC / 2 (44) |

Example

```
Option Public
```

```
Usesx "xlslc"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim num1 As New LCNumeric
```

Add Method for LCNumeric

This method adds the values of two LCNumeric objects.

Defined In
LCNumeric

Syntax
Call *numericTotal.Add(numeric1, numeric2)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|---------------------------------------|
| numeric1 | The first of two values being added. |
| numeric2 | The second of two values being added. |

Example

Option Public

Useslx "xlsxl"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim num1 As New LCNumeric

Dim num2 As New LCNumeric

Dim sum As New LCNumeric

num1.Value = 12345.6789

num2.Value = 12345.6789

Call sum.Add (Num1, Num2)

Print "The sum of the two numbers is " & sum.Text

End Sub

Example Output

The sum of the two numbers is 24691.3578

Compare Method for LCNumeric

This method compares two LCNumeric objects.

Defined In
LCNumeric

Syntax
result = *numericFirst*.**Compare**(*numericSecond*)

Parameters
numericSecond The base value to which to compare another value.

Return Value

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| <i>result</i> | Result of the comparison: Result > 0 (positive): numericFirst is greater than numericSecond. Result < 0 (negative): numericFirst is less than numericSecond. Result = 0: numericFirst is equal to numericSecond. |

Example

```
Option Public

Uselstx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim num1 As New LCNumeric
    Dim num2 As New LCNumeric

    num1.Value = 123.456789
    num2.Value = 123

    If (num1.Compare (num2) = 0) Then
        Print "The first number is the same as the second."
    ElseIf (num1.Compare (num2) > 0) Then
        Print "The first number is greater than the second."
    Else
        Print "The first number is less than the second."
    End If

End Sub
```

Example Output

The first number is greater than the second.

Copy Method for LCNumeric

This method makes a copy of one LCNumeric, creating a new LCNumeric object.

Defined In

LCNumeric

Syntax

Set *newNumeric* = *srcFirst*.Copy

Parameters

srcNumeric LCNumeric. The LCNumeric object that you want to copy.

Return Value

newNumeric LCNumeric. The copy of the *srcNumeric* object.

Example

Option Public

Useslx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim num1 As New LCNumeric

Dim num2 As LCNumeric

num1.Value = 12345.6789

Set num2 = num1.Copy

Print "The copy has a value of " & num2.Text

End Sub

Example Output

The copy has a value of 12345.6789

Subtract Method for LCNumeric

This method subtracts the value of one LCNumeric object from the value of another LCNumeric object.

Defined In
LCNumeric

Syntax

Call *numericThird.Subtract(numericFirst, numericSecond)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>numericFirst</i> | The initial value from which to subtract numericSecond. |
| <i>numericSecond</i> | The value to subtract from numericFirst. |

Example

Option Public

Uselstx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim num1 As New LCNumeric

Dim num2 As New LCNumeric

Dim diff As New LCNumeric

num1.Value = 98765.4321

num2.Value = 12345.6789

Call diff.Subtract (Num1, Num2)

Print "The difference of the two numbers is " & diff.Text

End Sub

Example Output

The difference of the two numbers is 86419.7532

Chapter 8

LCSession Class

This chapter provides information about Lotus Connectors LCSession class methods and properties.

Overview

The LCSession class represents the Lotus Connectors environment of the current script, providing access to the available connectors and metaconnectors, as well as the current error status. The status property and methods are useful when writing error handling code and reporting errors as text messages.

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session
- LCCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

```
Options
```

```
    Uselsx "*lsxlei"
```

```
Sub Initialize
```

```

Dim MyLEISession As New LCSession ("MainSession")

Dim MyNotesConnection as New LCConnection ("MyNotesConn")

Dim MyOracleConnection as New LCConnection
    ("MyOracleConn")

```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

LCSession Class Properties Summary

| <i>Property</i> | <i>Description</i> |
|------------------------------------|---|
| <i>LCSession.Status</i> | Status of an LCSession. Zero, or LCSUCCESS, indicates no error. A non-zero value (represented by an LCFAIL_XXX constant) indicates an error state. |
| <i>LCSession.ConnectionPooling</i> | Boolean. When set to FALSE (default), subsequently created connections are not pooled when the connection object is destroyed. Setting this property to TRUE enables connection pooling. For more information about Connection Pooling, refer to the Introduction in this manual. |

All Sessions have a set of predefined properties with values that can be assigned and retrieved. All properties are represented by both an integer value (called a property token) and a name.

Note LEI users can use these pre-defined properties with LCSession.GetProperty and LCSession.SetProperty methods, see Appendix E in the Lotus Enterprise Integrator User Guide for more information.

| <i>Property</i> | <i>Name</i> | <i>Description</i> |
|------------------------------|----------------|--|
| <i>LCTOKEN_NAME</i> | “Name” | Connector name. |
| <i>LCTOKEN_EVENT_ERROR</i> | “EventError” | Lotus Connector status code to treat as a non-error event. See below. |
| <i>LCTOKEN_TEXT_FORMAT</i> | “TextFormat” | Stream format constant for text data passed between the connector and the external system. This property is generally read-only. |
| <i>LCTOKEN_CHARACTER_SET</i> | “CharacterSet” | Lotus Connector character set indicator. See below. |
| <i>LCTOKEN_IGNORE_ERROR</i> | “IgnoreError” | Lotus Connector status code to ignore. See below. |

continued

| <i>Property</i> | <i>Name</i> | <i>Description</i> |
|-------------------------------|-----------------|---|
| <i>LCTOKEN_LCX_VERSION</i> | "LCXVersion" | LCX version (from LCXIDENTIFY structure). |
| <i>LCTOKEN_CONNECTOR_NAME</i> | "ConnectorName" | Convenience property assigned to the first sub-connector for a metaconnector LCX. Setting this property to a connector name is equivalent to creating a connector of this type and setting the connector itself to the subconnector property. |
| <i>LCTOKEN_IS_CONNECTED</i> | "IsConnected" | TRUE following a successful Connect call, until Disconnect is called. FALSE otherwise. |

Session properties may also be accessed by name. For example, the following line of script retrieves the value of the Text format for the platform:

```
char_set = session.TextFormat
```

Note DECS users can also access the property tokens using the above approach.

LCSession Class Methods Summary

This section lists the methods that are available to both DECS and LEI users.

There are several additional LCSession methods available only to LEI users. See Appendix E for detailed information.

| <i>Method</i> | <i>Description</i> |
|------------------------------------|--|
| <i>LCSession.ClearStatus</i> | Clear the Session status, putting the Session in a non-error state. |
| <i>LCSession.GetStatus</i> | Obtain the current Session status. |
| <i>LCSession.GetStatusText</i> | Obtain the error text string corresponding to a status code. |
| <i>LCSession.ListConnector</i> | List through all installed connectors available for LotusScript Extension for Lotus Connectors. |
| <i>LCSession.ListMetaConnector</i> | List through all installed metaconnectors available for a LotusScript Extension for Lotus Connectors installation. |

continued

| <i>Method</i> | <i>Description</i> |
|--------------------------------------|---|
| <i>LCSession.LookupConnector</i> | Determines if a specified connector is available. |
| <i>LCSession.LookupMetaConnector</i> | Determines if a specified metaconnector is available. |
| <i>LCSession.Sleep</i> | Suspends script execution for a specified period of time. |

New Method for LCSession

This is the constructor method for the LCSession object.

Multiple LCSession objects may be created within a script. All LCSession objects within a single script execution share the same status information.

Defined In

LCSession

Syntax

Dim variableName As New LCSession

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
```

ClearStatus Method for LCSession

This method resets, or clears, the LCSession status after an error.

Defined In

LCSession

Syntax

thisSession.ClearStatus()

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
```

```

Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
    REM Ignore errors
    On Error Resume Next

    REM purposely generate a Lotus Connector error
    session.ListConnector (23)

    Print "The current status code is #" & Cstr (session.Status)
    session.ClearStatus

    Print "The new status code is #" & Cstr (session.Status)

End Sub

```

Example Output

```

The current status code is #12292

The new status code is #0

```

GetStatus Method for LCSession

This method retrieves the current status value of a session.

Defined In
LCSession

Syntax

Call *thisSession.GetStatus (statusText, externalCode, externalText)*

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>statusText</i> | String. Optional. Lotus Connector status message. |
| <i>externalCode</i> | Long. Optional. Any external error codes. Long. |
| <i>externalText</i> | String. Optional. Any external message text associated with an external code. |

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    On Error Goto handler

```

```

Dim session As New LCSession

Dim src As New LCConnection ("db2")
  //For LEI syntax, see overview.

src.Database = "Gold"

src.UserID = "JDoe"

REM deliberate bad password

src.Password = "xyzyz"

src.Connect

Print "Connected to DB2."

End

handler:

If (session.Status <> LCSUCCESS) Then

  Dim text As String

  Dim extcode As Long

  Dim exttext As String

  Call session.GetStatus (text, extcode, exttext)

  If (session.Status = LCFAIL_EXTERNAL)Then

    Print "DB2 message: " & exttext & " code #" &
      Cstr(extcode)

  Else

    Print "Connector message: " & text

  End If

Else

  Print Error$

End If

End

End Sub

```

Example Output

```

DB2 message: [IBM] [CLI Driver] SQL1403N The username and/or
password supplied is incorrect. SQLSTATE=08004 code #-1403

```

GetStatusText Method for LCSession

This method returns the message text corresponding to an LCStatus code.

Defined In
LCSession

Syntax
message = *thisSession.GetStatusText*(errorCode)

Parameters
errorCode Long. Optional. The error code for which to return the message text. The default is the current status code.

Return Values
message A text string representation of the status.

Example
Option Public

```
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
    REM Ignore errors
    On Error Resume Next

    REM purposely generate a Lotus Connector error
    session.ListConnector (23)

    If (session.Status <> LCSUCCESS) Then
        Print session.GetStatusText
    Else
        Print "No error exists."
    End If
End Sub
```

Example Output
Error: Invalid List direction

ListConnector Method for LCSession

This method lists through all valid connectors for a Lotus Extension for Lotus Connectors installation.

The identifyFlagList and identifyNameList parameters provide the ability to obtain information from a Lotus Connector about its supported functionality and naming used by the back-end system.

Defined In
LCSession

Syntax
Call *thisSession.ListConnector(List, connectorName, connectorCode, identifyFlagList, identifyNameList)*

Parameters

| Value | Description |
|-------------------------|--|
| <i>list</i> | Long constant indicating whether to return the first or next Connector property. LCLIST_FIRST — Return the first property in the property list. LCLIST_NEXT — Return the next property (or the first property if this is the first call to this function for this Connection). |
| <i>connectorName</i> | String. Optional. String representation of the connector name. |
| <i>connectorCode</i> | String. Optional. The assigned Connector code. |
| <i>identifyFlagList</i> | Stream. Optional. IdentifyFlagList is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags (default is Nothing): LCIDFLAG_INDEX_CONNECTOR — Connector flags LCIDENTIFYF_XXX LCIDFLAG_INDEX_ACTION — Support actions for LCCConnection.Action method LCACTIDENTF_XXX LCIDFLAG_INDEX_OBJECT_CATALOG — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_CREATE — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_DROP — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| | <p>The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags ORed together:</p> <p>Supported Connector flags:</p> <p>LCIDENTIFYF_SINGLE_THREAD — Connector is not thread safe. The LSX LC will properly serialize access to this connector to avoid threading problems.</p> <p>LCIDENTIFYF_ARRAY_READ — Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.</p> <p>LCIDENTIFYF_ARRAY_WRITE — Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.</p> <p>LCIDENTIFYF_SINGLE_METADATA — All data is represented by a single metadata (for example, the File connector has only one metadata description).</p> <p>LCIDENTIFYF_WRITEBACKWriteback functionality is available.</p> <p>LCIDENTIFYF_SCROLLING — Scrolling result sets are available (currently not supported by any connectors).</p> <p>LCIDENTIFYF_MULTI_VALUE — Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the back-end.</p> <p>LCIDENTIFYF_MULTI_DIMENSION — Multi-dimensional result sets are supported (nested fieldlists).</p> <p>LCIDENTIFYF_SQL — SQL is the back-end-supported syntax.</p> <p>LCIDENTIFYF_SRVDB_CAT_CONNECT — Database connection is required for server and/or database browsing.</p> <p>LCIDENTIFYF_DISABLE_WRITEBACK — (Metaconnector only) The use of this metaconnector does not allow writeback result sets.</p> <p>Supported action flags:</p> <p>LCACTIDENTF_RESET — Reset action is supported.</p> <p>LCACTIDENTF_TRUNCATE — Truncate action is supported.</p> <p>LCACTIDENTF_COMMIT — Commit action is supported.</p> <p>LCACTIDENTF_ROLLBACK — Rollback action is supported.</p> <p>LCACTIDENTF_CLEAR — Clear action is supported.</p> <p>LCACTIDENTF_WAIT — Wait action is supported.</p> |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| | <p>Supported object flags:</p> <p>LCOBJIDENTF_SERVER — The method supports server objects.</p> <p>LCOBJIDENTF_DATABASE — The method supports database objects.</p> <p>LCOBJIDENTF_METADATA — The method supports metadata objects.</p> <p>LCOBJIDENTF_PROCEDURE — The method supports procedure objects.</p> <p>LCOBJIDENTF_INDEX — The method supports index objects.</p> <p>LCOBJIDENTF_FIELD — The method supports field objects.</p> <p>LCOBJIDENTF_PARAMETER — The method supports parameter objects.</p> <p>LCOBJIDENTF_ALT_METADATA — The method supports alternate metadata objects.</p> <p>LCOBJIDENTF_ALT_FIELD — The method supports alternate metadata field objects.</p> |
| <i>identifyNameList</i> | <p>Stream. Optional. IdentifyNameList is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's back-end system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):</p> <p>LCIDNAME_INDEX_SERVER — Name for server objects in this back-end system.</p> <p>LCIDNAME_INDEX_DATABASE — Name for database objects in this back-end system.</p> <p>LCIDNAME_INDEX_USERID — Name for user ID objects in this back-end system.</p> <p>LCIDNAME_INDEX_PASSWORD — Name for password objects in this back-end system.</p> <p>LCIDNAME_INDEX_METADATA — Name for metadata objects in this back-end system.</p> <p>LCIDNAME_INDEX_PROCEDURE — Name for procedure objects in this back-end system.</p> <p>LCIDNAME_INDEX_INDEX — Name for index objects in this back-end system.</p> |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| | LCIDNAME_INDEX_FIELD — Name for metadata fields in this back-end system. |
| | LCIDNAME_INDEX_PARAMETER — Name for procedure parameters in this back-end system. |
| | LCIDNAME_INDEX_ALT_METADATA — Name for alternate metadata objects in this back-end system. |
| | LCIDNAME_INDEX_ALT_FIELD — Name for alternate metadata fields in this back-end. |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim session As New LCSession //For LEI syntax, see overview.

Dim conName As String

Dim text As String

REM list the connectors available

REM the parameters for connector code, identity flags, and

REM identity names are optional and omitted in this example

Call session.ListConnector(LCLIST_FIRST, conName)

text = conName

While session.ListConnector(LCLIST_NEXT, conName)

text = text + ", " + conName

Wend

Print "The usable Connectors are " & text

End Sub

Example Output

The usable Connectors are db2, notes, odbc2, oracle, sybase

ListMetaConnector Method for LCSession

This method lists all valid metaconnectors for a Lotus Connectors installation.

Defined In
LCSession

Syntax
Call *thisSession*.**ListMetaConnector**(*list*, *metaconnectorName*, *connectorCode*, *identifyFlagList*, *identifyNameList*)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------------|---|
| <i>list</i> | Long. Constant indicating whether to return the first or next property: LCLIST_FIRST — Return the first property in the property list. LCLIST_NEXT — Return the next property (or the first property if this is the first call to this function for this Connection). |
| <i>metaconnectorName</i> | String. Optional. The name of the metaconnector. |
| <i>connectorCode</i> | String. Optional. The code for this metaconnector. |
| <i>identifyFlagList</i> | LCStream. Optional. IdentifyFlagList is set to a stream-of-format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags: LCIDFLAG_INDEX_CONNECTOR — Connector flags LCIDENTIFYF_XXX LCIDFLAG_INDEX_ACTION — Support actions for LCConnection.Action method LCACTIDENTF_XXX SLCIDFLLAG_INDEX_OBJECT_CATALOG — Support objects for LCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_CREATE — Support objects for LCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_DROP — Support objects for LCConnection.Catalog method LCOBJIDENTF_XXX |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| | <p>The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags ORed together:</p> <p>Supported Connector flags:</p> <p>LCIDENTIFYF_SINGLE_THREAD — Connector is not thread safe. The LSX will properly serialize access to this connector to avoid threading problems.</p> <p>LCIDENTIFYF_ARRAY_READ — Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.</p> <p>LCIDENTIFYF_ARRAY_WRITE — Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.</p> <p>LCIDENTIFYF_SINGLE_METADATA — All data is represented by a single metadata (for example, the File connector has only one metadata description).</p> <p>LCIDENTIFYF_WRITEBACK — Writeback functionality is available.</p> <p>LCIDENTIFYF_SCROLLING — Scrolling result sets are available (currently not supported by any connectors).</p> <p>LCIDENTIFYF_MULTI_VALUE — Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the back-end.</p> <p>LCIDENTIFYF_MULTI_DIMENSION — Multi-dimensional result sets are supported (nested fieldlists).</p> <p>LCIDENTIFYF_SQL — SQL is the backend-supported syntax.</p> <p>LCIDENTIFYF_SRVDB_CAT_CONNECT — Database connection is required for server and/or database browsing.</p> <p>LCIDENTIFYF_DISABLE_WRITEBACK — (Metaconnector only) The use of this metaconnector does not allow writeback result sets.</p> <p>Supported action flags:</p> <p>LCACTIDENTF_RESET — Reset action is supported.</p> <p>LCACTIDENTF_TRUNCATE — Truncate action is supported</p> <p>LCACTIDENTF_COMMIT — Commit action is supported.</p> <p>LCACTIDENTF_ROLLBACK — Rollback action is supported.</p> <p>LCACTIDENTF_CLEAR — Clear action is supported.</p> |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| | <p>LCACTIDENTF_WAIT — Wait action is supported.</p> <p>Supported object flags:</p> <p>LCOBJIDENTF_SERVER — The method supports server objects.</p> <p>LCOBJIDENTF_DATABASE — The method supports database objects.</p> <p>LCOBJIDENTF_METADATA — The method supports metadata objects.</p> <p>LCOBJIDENTF_PROCEDURE — The method supports procedure objects.</p> <p>LCOBJIDENTF_INDEX — The method supports index objects.</p> <p>LCOBJIDENTF_FIELD — The method supports field objects.</p> <p>LCOBJIDENTF_PARAMETER — The method supports parameter objects.</p> <p>LCOBJIDENTF_ALT_METADATA — The method supports alternate metadata objects.</p> <p>LCOBJIDENTF_ALT_FIELD — The method supports alternate metadata field objects.</p> |
| <i>identifyNameList</i> | <p>LCStream. Optional. IdentifyNameList is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's back-end system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):</p> <p>LCIDNAME_INDEX_SERVER — Name for server objects in this back-end system.</p> <p>LCIDNAME_INDEX_DATABASE — Name for database objects in this back-end system.</p> <p>LCIDNAME_INDEX_USERID — Name for user ID objects in this back-end system.</p> <p>LCIDNAME_INDEX_PASSWORD — Name for password objects in this back-end system.</p> <p>LCIDNAME_INDEX_METADATA — Name for metadata objects in this back-end system.</p> |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| | LCIDNAME_INDEX_PROCEDURE — Name for procedure objects in this back-end system. |
| | LCIDNAME_INDEX_INDEX — Name for index objects in this back-end system. |
| | LCIDNAME_INDEX_FIELD — Name for metadata fields in this back-end system. |
| | LCIDNAME_INDEX_PARAMETER — Name for procedure parameters in this back-end system. |
| | LCIDNAME_INDEX_ALT_METADATA — Name for alternate metadata objects in this back-end system. |
| | LCIDNAME_INDEX_ALT_FIELD — Name for alternate metadata fields in this back-end. |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax."

Sub Initialize

Dim session As New LCSession //For LEI syntax, see overview.

Dim conName As String

Dim text As String

REM list the connectors available

REM the parameters for connector code, identity flags, and

REM identity names are optional and omitted in this example

Call session.ListMetaConnector(LCLIST_FIRST, conName)

text = conName

While session.ListMetaConnector(LCLIST_NEXT, conName)

text = text + ", " + conName

Wend

Print "The usable MetaConnector(s) are " & text

End Sub

Example Output

The usable MetaConnector(s) are collexp, order

LookupConnector Method for LCSession

This method looks up a connector name to determine if it exists and returns information about the connector.

Defined In
LCSession

Syntax
Call *thisSession.LookupConnector*(*connectorName*, *connectorCode*, *identifyFlagList*, *identifyNameList*)

Parameters

| <i>Value</i> | <i>Description</i> |
|-------------------------|---|
| <i>connectorName</i> | String. The name of the Connector to look up, for example, "oracle." |
| <i>connectorCode</i> | Long. Optional. The Connector code for this Connector. |
| <i>identifyFlagList</i> | LCStream. Optional. IdentifyFlagList is set to a stream-of-format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags: LCIDFLAG_INDEX_CONNECTOR — Connector flags LCIDENTIFYF_XXX LCIDFLAG_INDEX_ACTION — Support actions for LCCConnection.Action method LCACTIDENTF_XXX LCIDFLAG_INDEX_OBJECT_CATALOG — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_CREATE — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX LCIDFLAG_INDEX_OBJECT_DROP — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags ORed together: Supported Connector flags: LCIDENTIFYF_SINGLE_THREAD — Connector is not thread safe. The LSX will properly serialize access to this connector to avoid threading problems. |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| | LCIDENTIFYF_ARRAY_READ — Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported. |
| | LCIDENTIFYF_ARRAY_WRITE — Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported. |
| | LCIDENTIFYF_SINGLE_METADATA — All data is represented by a single metadata (for example, the File connector has only one metadata description). |
| | LCIDENTIFYF_WRITEBACK — Writeback functionality is available. |
| | LCIDENTIFYF_SCROLLING — Scrolling result sets are available (currently not supported by any connectors). |
| | LCIDENTIFYF_MULTI_VALUE — Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the back-end. |
| | LCIDENTIFYF_MULTI_DIMENSION — Multi-dimensional result sets are supported (nested fieldlists). |
| | LCIDENTIFYF_SQL — SQL is the backend-supported syntax. |
| | LCIDENTIFYF_SRVDB_CAT_CONNECT — Database connection is required for server and/or database browsing. |
| | LCIDENTIFYF_DISABLE_WRITEBACK — (Metaconnector only) The use of this metaconnector does not allow writeback result sets. |
| | Supported action flags: |
| | LCACTIDENTF_RESET — Reset action is supported. |
| | LCACTIDENTF_TRUNCATE — Truncate action is supported. |
| | LCACTIDENTF_COMMIT — Commit action is supported. |
| | LCACTIDENTF_ROLLBACK — Rollback action is supported. |
| | LCACTIDENTF_CLEAR — Clear action is supported. |
| | LCACTIDENTF_WAIT — Wait action is supported. |
| | Supported object flags: |
| | LCOBJIDENTF_SERVER — The method supports server objects. |
| | LCOBJIDENTF_DATABASE — The method supports database objects. |
| | LCOBJIDENTF_METADATA — The method supports metadata objects. |
| | LCOBJIDENTF_PROCEDURE — The method supports procedure objects. |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| | LCOBJIDENTF_INDEX — The method supports index objects. |
| | LCOBJIDENTF_FIELD — The method supports field objects. |
| | LCOBJIDENTF_PARAMETER — The method supports parameter objects. |
| | LCOBJIDENTF_ALT_METADATA — The method supports alternate metadata objects. |
| | LCOBJIDENTF_ALT_FIELD — The method supports alternate metadata field objects. |
| <i>identifyNameList</i> | <p>LCStream. Optional. IdentifyNameList is set to a stream of format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's back-end system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing):</p> <p>LCIDNAME_INDEX_SERVER — Name for server objects in this back-end system.</p> <p>LCIDNAME_INDEX_DATABASE — Name for database objects in this back-end system.</p> <p>LCIDNAME_INDEX_USERID — Name for user ID objects in this back-end system.</p> <p>LCIDNAME_INDEX_PASSWORD — Name for password objects in this back-end system.</p> <p>LCIDNAME_INDEX_METADATA — Name for metadata objects in this back-end system.</p> <p>LCIDNAME_INDEX_PROCEDURE — Name for procedure objects in this back-end system.</p> <p>LCIDNAME_INDEX_INDEX — Name for index objects in this back-end system.</p> <p>LCIDNAME_INDEX_FIELD — Name for metadata fields in this back-end system.</p> <p>LCIDNAME_INDEX_PARAMETER — Name for procedure parameters in this back-end system.</p> <p>LCIDNAME_INDEX_ALT_METADATA — Name for alternate metadata objects in this back-end system.</p> <p>LCIDNAME_INDEX_ALT_FIELD — Name for alternate metadata fields in this back-end.</p> |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim session As New LCSession //For LEI syntax, see overview.

REM the optional parameters have been omitted in this example

If (session.LookupConnector ("db2")) Then

Print "DB2 connectivity installed."

Else

Print "DB2 connectivity is not installed."

End If

End Sub

Example Output

DB2 connectivity installed.

LookupMetaConnector Method for LCSession

This method looks up a metaconnector name to determine if it exists and returns information about the metaconnector.

Defined In

LCSession

Syntax

Call *thisSession.LookupMetaConnector*(*metaconnectorName*, *connectorCode*, *tokenBase*, *identifyFlagList*, *identifyNameList*)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------------|---|
| <i>metaconnectorName</i> | String. The name of the metaconnector to look up. |
| <i>connectorCode</i> | Long. Optional. The Connector code of the metaconnector. |
| <i>tokenBase</i> | Long. Optional. The tokenbase for this metaconnector. |
| <i>identifyFlagList</i> | <p>LCStream. Optional. IdentifyFlagList is set to a stream of format number list (LCSTREAMFMT_NUMBER_LIST), which will contain a series of flags from the connector. Use the NumberListGetValue method of LCStream to retrieve individual flag values as required. The specific flags to retrieve are indicated by the index in the number list. Use the following constants to represent a particular set of flags:</p> <p>LCIDFLAG_INDEX_CONNECTOR — Connector flags LCIDENTIFYF_XXX</p> <p>LCIDFLAG_INDEX_ACTION — Support actions for LCCConnection.Action method LCACTIDENTF_XXX</p> <p>SLCIDFLAG_INDEX_OBJECT_CATALOG — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX</p> <p>LCIDFLAG_INDEX_OBJECT_CREATE — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX</p> <p>LCIDFLAG_INDEX_OBJECT_DROP — Support objects for LCCConnection.Catalog method LCOBJIDENTF_XXX</p> <p>The following are the valid flag values. Each flag set is composed of zero or more of the corresponding flags ORed together:</p> <p>Supported Connector flags:</p> <p>LCIDENTIFYF_SINGLE_THREAD — Connector is not thread safe. The LSX will properly serialize access to this connector to avoid threading problems.</p> <p>LCIDENTIFYF_ARRAY_READ — Array reads (more efficient handling of RecordCount > 1 for Fetch method) are supported.</p> <p>LCIDENTIFYF_ARRAY_WRITE — Array writes (more efficient handling of RecordCount > 1 for Insert, Update, and/or Remove methods) are supported.</p> <p>LCIDENTIFYF_SINGLE_METADATA — All data is represented by a single metadata (for example, the File connector has only one metadata description).</p> <p>LCIDENTIFYF_WRITEBACK — Writeback functionality is available.</p> |

continued

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| | LCIDENTIFYF_SCROLLING — Scrolling result sets are available (currently not supported by any connectors). |
| | LCIDENTIFYF_MULTI_VALUE — Multi-value types (Binary stream formats for text, number, and datetime lists) are fully supported by the back-end. |
| | LCIDENTIFYF_MULTI_DIMENSION — Multi-dimensional result sets are supported (nested fieldlists). |
| | LCIDENTIFYF_SQL — SQL is the backend-supported syntax. |
| | LCIDENTIFYF_SRVDB_CAT_CONNECT — Database connection is required for server and/or database browsing. |
| | LCIDENTIFYF_DISABLE_WRITEBACK — (Metaconnector only) The use of this metaconnector does not allow writeback result sets. |
| | Supported action flags: |
| | LCACTIDENTF_RESET — Reset action is supported. |
| | LCACTIDENTF_TRUNCATE — Truncate action is supported |
| | LCACTIDENTF_COMMIT — Commit action is supported. |
| | LCACTIDENTF_ROLLBACK — Rollback action is supported |
| | LCACTIDENTF_CLEAR — Clear action is supported. |
| | LCACTIDENTF_WAIT — Wait action is supported. |
| | Supported object flags: |
| | LCOBJIDENTF_SERVER — The method supports server objects. |
| | LCOBJIDENTF_DATABASE — The method supports database objects. |
| | LCOBJIDENTF_METADATA — The method supports metadata objects. |
| | LCOBJIDENTF_PROCEDURE — The method supports procedure objects. |
| | LCOBJIDENTF_INDEX — The method supports index objects. |
| | LCOBJIDENTF_FIELD — The method supports field objects. |
| | LCOBJIDENTF_PARAMETER — The method supports parameter objects. |
| | LCOBJIDENTF_ALT_METADATA — The method supports alternate metadata objects. |
| | LCOBJIDENTF_ALT_FIELD — The method supports alternate metadata field objects. |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| <i>identifyNameList</i> | <p>LCStream. Optional. IdentifyNameList is set to a stream-of-format text list (LCSTREAMFMT_TEXT_LIST), which will contain a series of names used by the connector's back-end system. This can be used to customize the presentation of options for a specific connector (for example, metadata is named "Form" for Notes, and "Table" for DB2). Use the TextListGetValue method of LCStream to retrieve individual names as required. The specific name to retrieve is indicated by the index in the text list. Use the following constants to represent a particular name (default is Nothing).</p> <p>LCIDNAME_INDEX_SERVER — Name for server objects in this back-end system.</p> <p>LCIDNAME_INDEX_DATABASE — Name for database objects in this back-end system.</p> <p>LCIDNAME_INDEX_USERID — Name for user ID objects in this back-end system.</p> <p>LCIDNAME_INDEX_PASSWORD — Name for password objects in this back-end system.</p> <p>LCIDNAME_INDEX_METADATA — Name for metadata objects in this back-end system.</p> <p>LCIDNAME_INDEX_PROCEDURE — Name for procedure objects in this back-end system.</p> <p>LCIDNAME_INDEX_INDEX — Name for index objects in this back-end system.</p> <p>LCIDNAME_INDEX_FIELD — Name for metadata fields in this back-end system.</p> <p>LCIDNAME_INDEX_PARAMETER — Name for procedure parameters in this back-end system.</p> <p>LCIDNAME_INDEX_ALT_METADATA — Name for alternate metadata objects in this back-end system.</p> <p>LCIDNAME_INDEX_ALT_FIELD — Name for alternate metadata fields in this back-end.</p> |

Example

```
Option Public
```

```
Useslx "xlsxl"
```

```
// DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
Dim session As New LCSession //For LEI syntax, see overview.
```

```
REM the optional parameters have been ommited in this example
```

```

If (session.LookupMetaConnector ("order")) Then
    Print "The 'order' meta connector is present."
Else
    Print "The 'order' meta connector is not present."
End If
End Sub

```

Example Output

The 'order' meta connector is present.

Sleep Method for LCSession

This method forces a script execution to sleep for the specified length of time.

Defined In

LCSession

Syntax

Call *thisSession.Sleep*(*milliseconds*)

Parameters

milliseconds Long. Number of milliseconds to sleep.

Example

```

Option Public

Uselstx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim session As New LCSession //For LEI syntax, see overview.
    Print "The Time is " Cstr(Now)
    session.Sleep (5000)
    Print "The Time is " Cstr(Now)
End Sub

```

Example Output

The Time is 9/8/98 5:23:32 PM

The Time is 9/8/98 5:23:37 PM

Chapter 9

LCStream Class

This chapter provides information about the Lotus Connector LCStream class methods and properties.

Overview

The LCStream class represents text and binary datatypes. A stream value is a variable-length list of characters or bytes. Streams come in two basic types, text and binary, represented by the format of the stream. Specific format information indicates either the character set (for text) or the binary format (for binary).

In addition to specific text formats, there is also the option to designate a text stream as “Native,” or LCSTREAMFMT_NATIVE, indicating the characters of the stream should be stored in the local platform specific character set. (Note that the LSX LC uses unicode for text representation. To create a unicode stream, use LCSTREAMFMT_UNICODE.) Likewise, in addition to the basic binary designation (BLOB or LCSTREAMFMT_BLOB), there are four specialized binary formats:

- LCSTREAMFMT_COMPOSITE — Notes composite (Notes Rich Text format)
- LCSTREAMFMT_TEXT_LIST — list of LMBCS text strings
- LCSTREAMFMT_NUMBER_LIST — list of double precision floating point values and ranges
- LCSTREAMFMT_DATETIME_LIST — list of LCDatetime values and ranges

There are special methods dedicated to working with the three “LIST” formats. (The maximum storage size of the “LIST” format stream object is 64K.)

For LEI Users: Understanding the Uselsx Statement in the Examples

The Uselsx statement at the beginning of each LSX program indicates which LSX API to use.

Each method described in this manual is accompanied by basic example syntax. Typically, these examples show the following statement, which calls the LSX LC API used by DECS:

```
Uselsx "*lsxlc"
```

As of LEI Release 3.1, the LEI LSX uses the LSX LEI API. If you are writing LSX programs to be executed in the LEI environment, you must use the following statement:

```
Uselsx "*lsxlei"
```

In addition, do the following:

- Dim LCSession as a named session
- LCConnections must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

```
Options
    Uselsx "*lsxlei"
Sub Initialize
    Dim MyLEISession As New LCSession ("MainSession")
    Dim MyNotesConnection as New LCConnection ("MyNotesConn")
    Dim MyOracleConnection as New LCConnection
        ("MyOracleConn")
```

Note “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

LCStream Type TEXT Format and Values

The following table shows the format and values for the TEXT type.

| <i>Description</i> | <i>Value</i> |
|--------------------|--|
| Type constant | LCTYPE_TEXT |
| Description | Locale-specific character stream |
| Other | Maximum Length = 4 Gb Formats: LMBCS, Native, or any valid character set (see Appendix D) |

LCStream Type BINARY Format and Values

The following table show format and values for the BINARY type.

| <i>Description</i> | <i>Values</i> |
|--------------------|--|
| Type constant | LCTYPE_BINARY |
| Value | LCSTREAM structure |
| Description | Optionally formatted byte stream |
| Other | Maximum Length = 4 Gb Formats: BLOB (unformatted), Notes rich text, Notes text list, Notes number list, Notes datetime list |

A stream value contains the following information:

| <i>Value</i> | <i>Description</i> |
|----------------|--|
| Maximum Length | The maximum valid data length for this stream. Any value is valid, with a value of zero indicating no maximum length. |
| Stream Flags | Zero or more stream flags ORed together. See Stream Flags description below. |
| Stream Format | Stream data format. See Stream Format description below. If stream format is not a list type, it is a one-element string or a binary stream. |
| Data Length | Length, in bytes, of data in the data buffer field. |

Stream Flags

The StreamFlags of a stream determine the behavior of the stream under particular circumstances:

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| LCSTREAMF_FIXED | Buffer length is always MaxLength, and the buffer is allocated once and never changed. MaxLength cannot be zero. Without this flag, the buffer is dynamically reallocated to accommodate assigned values (within MaxLength). |
| LCSTREAMF_TRUNCATE | Stream will automatically truncate assigned and converted values if the stream cannot accommodate them. Without this flag, assigning a value too long for the stream will generate an overflow error. |

continued

| <i>Value</i> | <i>Description</i> |
|-------------------|---|
| LCSTREAMF_NO_CASE | Stream is not case sensitive. During text stream comparisons, a non case sensitive comparison is done when either or both streams have this flag set. Without this flag, text-to-text comparisons are case sensitive. |
| LCSTREAMF_NO_TRIM | Stream should not be trimmed. This will notify Connectors and the Trim method that trimming of trailing spaces should not be performed for this stream. |

Stream Format

The Format of a stream indicates the structure of the stream data. The flag LCSTREAMFMTF_BINARY indicates whether the stream is of a binary format or a text format. Text formats are represented by either a character set constant or the following special value:

LCSTREAMFMT_NATIVE

The same as the native character set of the local machine.

For a complete list of supported character sets, see Appendix D.

Binary formats must be one of the following values:

| <i>Value</i> | <i>Description</i> |
|----------------------------------|---|
| <i>LCSTREAMFMT_BLOB</i> | Unformatted (Binary Large Object) |
| <i>LCSTREAMFMT_COMPOSITE</i> | Lotus Notes format Composite (also known as Rich Text or Compound Text) |
| <i>LCSTREAMFMT_TEXT_LIST</i> | Lotus Notes format Text List (multi-value list of text values) |
| <i>LCSTREAMFMT_NUMBER_LIST</i> | Lotus Notes format Number List (multi-value list of number values and ranges) |
| <i>LCSTREAMFMT_DATETIME_LIST</i> | Lotus Notes format Datetime List (multi-value list of datetime values and ranges) |

Conversion is supported between stream formats excluding certain conversions. If composite is involved in a conversion, it must be the source and the target must be BLOB or a text format. In addition, conversion between number list and datetime list is not supported.

Stream Buffer

The maximum length of a stream indicates the maximum valid length in bytes for a value assigned to this stream. This can be any value up to 4 Gb.

A value of zero indicates that the stream has no maximum length.

LCStream Properties

| <i>Value</i> | <i>Description</i> |
|-------------------|---|
| <i>Flags</i> | Long. The flags for the stream. |
| <i>Format</i> | Long. The stream data format. |
| <i>Length</i> | Long. The length of the stream data. |
| <i>MaxLength</i> | Long. The maximum valid data length for the stream. Any value is valid, with a value of zero indicating no maximum length. [Read-Only] |
| <i>Text</i> | String. Text representation of the stream. |
| <i>Value</i> | Variant. An array. If contents of Stream is a string, it's a one element array. If numbers or datetimes, it is an array of numbers. It can also be an array of strings for TextLists. |
| <i>ValueCount</i> | Long. Number of elements in a stream. Valid only for List<Type> formats. [Read-Only] |
| <i>RangeCount</i> | Long. Range of elements in a stream. Valid only for List<Type> formats. [Read-Only] |

New Method for LCStream

This is the constructor method for the LCStream class. It creates an empty LCStream object and optionally assigns initial properties.

Defined In
LCStream

Syntax

Dim variableName As New LCStream(*maxLength*, *flags*, *format*)

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| <i>maxLength</i> | Long. Optional. Maximum length that this stream's data can be. A value of zero indicates no maximum, which is not valid with the flag LCSTREAMF_FIXED. Default is 0. |
| <i>streamFlags</i> | Long. Optional. Flags for this stream. When using the flag LCSTREAMF_FIXED to create a fixed-length stream, the stream's data buffer is allocated to be maxLength bytes. Refer to the Stream Flags section for a description of the flags. The default is 0. |
| <i>format</i> | Long. Optional. Initial stream format to be assigned to the stream. The default is LCSTREAMFMT_UNICODE. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim message As New LCStream
```

Clear Method for LCStream

This method clears a stream value and properties.

Defined In

LCStream

Syntax

lcStream.Clear

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim message As New LCStream

    message.Text = "Hello World"

    Message.Clear

    If (Len(message.Text) > 0) Then

        Print "the message is " & message.Text

    Else

        Print "the message is blank"

    End If

End Sub
```

Example Output

the message is blank

Append Method for LCStream

This method appends one stream to another to yield a third LCStream object containing the data from both.

Defined In
LCStream

Syntax

Call *newStream.Append(stream1, stream2)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------|--|
| <i>stream1</i> | LCStream. The stream to which you want to append. The newStream will have this stream's format. |
| <i>stream2</i> | LCStream. The stream to append to stream1. If this stream is a different format than stream1, it will first be converted to the format of stream1. |

Example

```
Option Public
Option Explicit
Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim stream1 As New LCStream (64, , LCSTREAMFMT_ASCII)
    Dim stream2 As New LCStream (64, , LCSTREAMFMT_ASCII)
    Dim together As New LCStream (64, , LCSTREAMFMT_ASCII)
    stream1.Text = "The quick brown fox "
    stream2.Text = "jumped over the lazy dog."
    Call together.Append (stream1, stream2)
    Print "The combined stream is " & together.Text
End Sub
```

Example Output

The combined stream is The quick brown fox jumped over the lazy dog.

Compare Method for LCStream

This method compares two LCStream objects to determine if they are equal or unequal. If the streams are of different formats, they will first be converted to the same format. If both streams are text, they may be converted to unicode for comparison. In a text comparison, if either stream has the flag LCSTREAMF_NO_CASE set, then a case-insensitive text comparison will be done.

Defined In
LCStream

Syntax
result = thisStream.Compare(baseStream)

Parameters
baseStream LCStream. The stream to which you want to compare this Stream.

| Return Value | |
|---------------|--|
| Value | Value |
| <i>result</i> | Result of the comparison: Result > 0 (positive): thisStream is greater than baseStream. Result < 0 (negative): thisStream is less than baseStream. Result = 0: thisStream is equal to baseStream. |

Example

```
Option Public
Option Explicit
Usesx "xlslc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize
    Dim stream1 As New LCStream (64, LCSTREAMF_NO_CASE,
        LCSTREAMFMT_ASCII)
    Dim stream2 As New LCStream (64, , LCSTREAMFMT_ASCII)
    Dim match As Long
    stream1.Text = "The quick brown fox"
    stream2.Text = "the QuICK BROWn FOX"
    match = stream1.Compare (stream2)
    If (match = 0) Then
```



```

        Print "The first string is the same as the second."
    Elseif (match > 0) Then
        Print "The first string comes before the second."
    Else
        Print "The first string comes after the second."
    End If
End Sub

```

Example Output

The first string is the same as the second.

Convert Method for LCStream

This method obtains a stream in a particular stream format.

Defined In
LCStream

Syntax

Call *newStream.Convert* (*srcStream*, *flags*, *format*)

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|---|
| <i>srcStream</i> | LCStream. Stream whose data is to be converted. |
| <i>flags</i> | <p>Long. Flags that determine aspects of the conversion. Zero or more of the following values, OR-ed together:</p> <p>LCCONVERTF_REFERENCE Allows the destination stream to reference the source stream if they are of the same format. Otherwise, a data copy is always required. See Usage Notes below.</p> <p>LCCONVERTF_PRESERVE Preserves the destination stream meta-information (MaxLength and StreamFlags). Otherwise, they are taken from the source stream.</p> <p>LCCONVERTF_FORCE_TEXT Overrides the DECSTranslation setting of 0 or 1 in the NOTES.INI file for text translation and forces it to occur between any different text character sets.</p> |
| <i>format</i> | Long. Stream format of the destination data. |

Usage Notes

The use of the flag `LCCONVERTF_REFERENCE` supports an efficient method of obtaining stream data in a particular format. When requesting a stream in a specific format and using this flag, if the stream data is already in the same format, no data copying is done. The new stream simply references the data of the original stream.

When working with a stream that references another stream's data, do delete the original stream until the referencing stream is no longer needed.

Example

Option Public

```
Usesx "lsxlc" // DECS only, see overview in this chapter for  
LEI syntax.
```

```
Sub Initialize
```

```
    Dim msga As New LCStream (0, 0, LCSTREAMFMT_ASCII)
```

```
    Dim msgu As New LCStream
```

```
    msga.Text = "Peter Pan lived in Never Never Land."
```

```
    Call msgu.Convert (msga, 0, LCSTREAMFMT_UNICODE)
```

```
    Print "The length of the msg in ASCII is " & msga.Length & "  
while the length in Unicode is " & msgu.Length
```

```
End Sub
```

Example Output

The length of the msg in ASCII is 36, while the length in Unicode is 72.

Copy Method for LCStream

This method copies the value one `LCStream` object to another.

Defined In

`LCStream`

Syntax

Set newStream = lcStream.Copy

Parameters

lcStream `LCStream`. Source stream to be copied.

Return Value

newStream `LCStream`. The copy of the original stream.

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim msg As New LCStream

    Dim newmsg As LCStream

    msg.Text = "Peter Pan"

    Set newmsg = msg.Copy

    Print "The original msg is " & msg.Text

    Print "The new msg is " & newmsg.Text

End Sub
```

Example Output

```
The original msg is Peter Pan

The new msg is Peter Pan
```

Extract Method for LCStream

This method creates a stream from part of the data of an existing stream.

Since this method works off of byte counts, it is most useful if you know you have single or double byte character sets. It's not as useful if there's a mix, like LMBCS or CodePage932 (mix of single-byte and double-byte character sets).

Defined In

LCStream

Syntax

Call *lcStream.Extract (srcStream, offset, length)*

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>srcStream</i> | LCStream. Stream supplying the data from which the new stream is created. |
| <i>offset</i> | Long. Byte position in SrcStream of the start of the new stream. If the offset exceeds the length of the stream data, the new stream is cleared. |
| <i>length</i> | Long. Length in bytes of the new stream copied from SrcStream. If the length copies more bytes than are available, the copy stops at the end of the source data. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim msg As New LCStream (, , LCSTREAMFMT_ASCII)

    Dim part As New LCStream

    msg.Text = "the quick brown fox jumped over the lazy dog"

    Call part.Extract (msg, 11, 5)

    Print "The 5 bytes, starting at the 11th byte is " &
        part.Text

End Sub
```

Example Output

The 5 bytes, starting at the 11th byte is brown

Merge Method for LCStream

This method combines one stream into another, producing a new stream.

Defined In

LCStream

Syntax

Call *lcStream.Merge*(*stream1*, *offset*, *stream2*)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------|--|
| <i>stream1</i> | LCStream. stream2 is combined into stream1. |
| <i>offset</i> | Long. Position in lcStream of the first byte of stream2: stream2 is inserted into stream1 starting at the position indicated by offset. |
| <i>stream2</i> | LCStream. stream2 is combined into stream1. If stream2 is a different format than stream1, it is converted to the format of stream1 before being merged. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize
```

```

Dim msg As New LCStream          ' Unicode is the default
    'format for streams

Dim part As New LCStream

Dim newmsg As New LCStream

msg.Text = "the quick brown fox jumped over the lazy dog"
part.Text = "very "

REM counting start with 1 and each character in a unicode
    string is 2 bytes

Call newmsg.Merge (msg, 9, part)

Print "The message, after inserting '" & part.Text"' starting
    at the 9th byte, is " & newmsg.Text

End Sub

```

Example Output

The message, after inserting 'very ' starting at the 9th byte,
is the very quick brown fox jumped over the lazy dog

ResetFormat Method for LCStream

This method resets the format an LCStream object, without affecting the data or other properties.

This method may be useful when storing multiple language strings in a single database and you need to switch the format. A typical case would be web browsers accessing an application from different countries and requiring different translations of the content. The different translations could be stored in a single database. Since most RDBMS environments do not permit multiple languages within a single database, the translations could be selected based on a key. The contents would be treated as blobs and the format 'reset' to the correct character set before being passed to the web browser.

Defined In
LCStream

Syntax
Call *lcStream.ResetFormat* (*format*)

Parameters

format Long. The new format for the stream object. See the list of stream formats in the section "Stream Formats" earlier in this chapter.

Example

Option Public

Usesx "*/sxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim msg As New LCStream (0, 0, LCSTREAMFMT_ASCII)

msg.Text = "Peter Pan lived in Never Never Land."

Print "The length of the msg is " & msg.Length

Call msg.ResetFormat (LCSTREAMFMT_UNICODE)

Print "The length of the msg and its contents have not
changed but the format is now Unicode"

Print "The length of the msg is " & msg.Length

End Sub

Example Output

The length of the msg is 36

The length of the msg and its contents have not changed but the
format is now Unicode

The length of the msg is 36

SetFormat Method for LCStream

This method sets the format for an LCStream object, converting the stream data if necessary. This method may be more efficient than the Convert method, since a second stream is not required.

Defined In

LCStream

Syntax

Call *lcStream.setFormat (format)*

Parameters

format Long. The format to which to set the stream object. See the list of stream formats in the section “Stream Formats” earlier in this chapter. The data will be converted to this stream format.

Example

Option Public

```

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim msg As New LCStream (0, 0, LCSTREAMFMT_ASCII)
    msg.Text = "Peter Pan lived in Never Never Land."
    Print "The length of the msg in ASCII is " & msg.Length
    Call msg.SetFormat (LCSTREAMFMT_UNICODE)
    Print "The length of the msg in Unicode is " & msg.Length
End Sub

```

Example Output

The length of the msg in ASCII is 36

The length of the msg in Unicode is 72

Trim Method for LCStream

This method trims trailing spaces from text streams. If the stream is not a text format or if the stream has the flag `LCSTREAM_NO_TRIM` set, then this method will perform no action.

Defined In

LCStream

Syntax

lcStream.Trim

Example

Option Public

```

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim msg As New LCStream

    msg.Text = "    this has space at the start and the end    "
    msg.Trim ' trim trailing spaces (not leading ones)
    Print "The msg is *" & msg.Text & "*"
End Sub

```

Example Output

The msg is * this has space at the start and the end*

DatetimeListGetRange Method for LCStream

This method gets a range of values in an LCStream DatetimeList object.
The stream must be of the proper format, i.e.,
LCSTREAMFMT_DATETIME_LIST.

Defined In
LCStream

Syntax
Call *lcStream.DatetimeListGetRange (index, startDatetime, endDatetime)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>index</i> | Long. Index position of the range in the datetimeList stream. |
| <i>startDatetime</i> | LCDatetime. Output. The starting datetime of the range. |
| <i>endDatetime</i> | LCDatetime. Output. The ending datetime of the range. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim datelist As New LCStream (0, 0,
        LCSTREAMFMT_DATETIME_LIST)

    Dim start As New LCDatetime
    Dim finish As New LCDatetime

    datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000
        12:00AM," & _ "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"

    Call datelist.DatetimeListGetRange (1, start, finish)

    Print "The new 1st range is " & start.Text & " - " &
        finish.Text

End Sub
```

Example Output
The new 1st range is 05:00:00 PM - 06:00:00 PM

DatetimeListGetValue Method for LCStream

This method retrieves a datetime value from a specified place in a DateTimeList LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In
LCStream

Syntax

Call *lcStream.DatetimeListGetValue(index, datetime)*

Parameters

| <i>Value</i> | <i>Description</i> |
|-----------------|---|
| <i>index</i> | Long. Index position of the datetime value to retrieve. |
| <i>datetime</i> | LCDatetime. The datetime value. |

Example

Option Public

Uselsx "**lsxlc*"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

 Dim datelist As New LCStream (0, 0,
 LCSTREAMFMT_DATETIME_LIST)

 Dim num As New LCDatetime

 datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000
 12:00AM," & _
 "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"

 Call datelist.DatetimeListGetValue (3, num)

 Print "The new 3rd datetime is " & num.Text

End Sub

Example Output

The new 3rd datetime is 12/31/1999 12:59:00 PM

DatetimeListInsertRange Method for LCStream

This method inserts inserts a datetime range into a datetime list stream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_DATETIME_LIST.

Defined In
LCStream

Syntax
Call *datetimeNew.DatetimeListInsertRange* (*index*, *startDatetime*, *endDatetime*)

Parameters

| Value | Description |
|----------------------|--|
| <i>index</i> | Long. The position at which to insert the value(s). |
| <i>startDatetime</i> | LCDatetime. The first datetime value for the range. |
| <i>endDatetime</i> | LCDatetime. The second datetime value for the range. |

Example
Option Public

```
Useslx "slsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim datelist As New LCStream (0, 0,
        LCSTREAMFMT_DATETIME_LIST)

    Dim start As New LCDatetime
    Dim finish As New LCDatetime

    start.Text = "8/6/1941"
    finish.Text = "7/1/1997"

    datelist.Text = "12/25/50, 7:00AM, 12/31/99 12:59PM, 1/1/2000
        12:00AM, 5:00PM - 6:00PM, 6:30AM, 5/1/96 - 5/31/96"

    Call datelist.DatetimeListInsertRange (2, start, finish)

    Print "The new stream is " & datelist.Text

End Sub
```

Example Output

The new stream is 12/25/1950, 07:00:00 AM, 12/31/1999 12:59:00 PM, 01/01/2000 12:00:00 AM, 06:30:00 AM, 05:00:00 PM - 06:00:00 PM, 08/06/1941 - 07/01/1997, 05/01/1996 - 05/31/1996

DatetimeListInsertValue Method for LCStream

This method inserts a value into a datetime list LCStream object.

The stream must be of the proper format, such as
LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListInsertValue* (*index*, *datetime*)

Parameters

| <i>Value</i> | <i>Description</i> |
|-----------------|--|
| <i>index</i> | Long. The index position at which to insert the value. |
| <i>datetime</i> | LCDatetime. The datetime value to insert. |

Example

Option Public

Usesx "xlslc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

```
Dim datelist As New LCStream (0, 0,  
    LCSTREAMFMT_DATETIME_LIST)
```

```
Dim clock As New LCDatetime
```

```
clock.SetCurrent
```

```
datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000  
    12:00AM," & _ "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
Call datelist.DatetimeListInsertValue (3, clock)
```

```
Print "The new stream is " & datelist.Text
```

End Sub

Example Output

```
The new stream is 12/25/1950, 07:00:00 AM, 09/08/1998
05:22:23.96 PM, 12/31/1999 12:59:00 PM, 01/01/2000 12:00:00 AM,
06:30:00 AM, 05:00:00 PM - 06:00:00 PM, 05/01/1996 - 05/31/1996
```

DatetimeListRemoveRange Method for LCStream

This method removes a range from a datetime list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimeListRemoveRange* (*index*)

Parameters

index Long. The position of the value to remove.

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

```
Dim datelist As New LCStream (0, 0,
    LCSTREAMFMT_DATETIME_LIST)
```

```
datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000
    12:00AM," & _"5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
Call datelist.DatetimeListRemoveRange (2)
```

```
Print "The new stream is " & datelist.Text
```

End Sub

Example Output

```
The new stream is 12/25/1950, 07:00:00 AM, 12/31/1999
12:59:00 PM, 01/01/2000 12:00:00 AM, 06:30:00 AM, 05:00:00 PM -
06:00:00 PM
```

DatetimestreamRemoveValue Method for LCStream

This method removes a value from a datetime list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_DATETIME_LIST.

Defined In

LCStream

Syntax

Call *lcStream.DatetimestreamRemoveValue(index)*

Parameters

index Long. The position of the value to remove.

Example

Option Public

Useslsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

```
Dim datelist As New LCStream (0, 0,  
    LCSTREAMFMT_DATETIME_LIST)
```

```
datelist.Text = "12/25/50,7:00AM,12/31/99 12:59PM,1/1/2000  
    12:00AM," & _ "5:00PM - 6:00PM,6:30AM,5/1/96 - 5/31/96"
```

```
Call datelist.DatetimestreamRemoveValue (3)
```

```
Print "The new stream is " & datelist.Text
```

End Sub

Example Output

```
The new stream is 12/25/1950, 07:00:00 AM, 01/01/2000 12:00:00  
AM, 06:30:00 AM, 05:00:00 PM - 06:00:00 PM, 05/01/1996 -  
05/31/1996
```

NumberListGetRange Method for LCStream

This method selects a particular range from a number list LCStream object.
The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In
LCStream

Syntax
Call *lcStream.NumberListGetRange (index, startNumber, endNumber)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| <i>index</i> | Long. The position of the range to select. |
| <i>startNumber</i> | Double. Output. The first number value for the range. |
| <i>endNumber</i> | Double. Output. The second number value for the range. |

Example

```
Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

    Dim start As Double

    Dim finish As Double

    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"

    Call numlist.NumberListGetRange (1, start, finish)

    Print "The new 1st range is " & start & " - " & finish

End Sub
```

Example Output
The new 1st range is 50 - 55

NumberListGetValue Method for LCStream

This method retrieves a specified value from a number list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In
LCStream

Syntax

Call *lcStream.NumberListGetValue* (*index, number*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------|--|
| <i>index</i> | Long. The index position of the value to retrieve. |
| <i>number</i> | Double. The variable in which to place the value. |

Example

Option Public

Uselsx "**lsxlc*"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

Dim num As Double

numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"

Call numlist.NumberListGetValue (3, num)

Print "The new 3rd number is " & num

End Sub

Example Output

The new 3rd number is 33

NumberListInsertRange Method for LCStream

This method inserts a number range into a number list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In
LCStream

Syntax

Call *lcStream.NumberListInsertRange(index, startNumber, endNumber)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------------|--|
| <i>index</i> | Long. The index position at which to insert the value. |
| <i>startNumber</i> | Double. The first number of the range. |
| <i>endNumber</i> | Double. The second number of the range. |

Example

Option Public

```
Useslx "lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.
Sub Initialize

    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)
    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"
    Call numlist.NumberListInsertRange (2, 8.8, 9.9)

    Print "The new stream is " & numlist.Text
End Sub
```

Example Output

The new stream is 1.11, 22.2, 33, 4.444, 66, 50 - 55, 8.8 - 9.9, 77 - 79

NumberListInsertValue Method for LCStream

This method inserts a value into a number list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream.NumberListInsertValue(index, number)*

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------|--|
| <i>index</i> | Long. The position at which to insert the value. |
| <i>number</i> | Double. The value to insert. |

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"

Call numlist.NumberListInsertValue (3, 99.99)

Print "The new stream is " & numlist.Text

End Sub

Example Output

The new stream is 1.11, 22.2, 99.99, 33, 4.444, 66, 50 - 55, 77
- 79

NumberListRemoveRange Method for LCStream

This method removes a range of numbers as indicated by an index from a number list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In

LCStream

Syntax

Call *lcStream*.NumberListRemoveRange(*index*)

Parameters

index Long. The position of the range to remove.

Example

Option Public

Uselsx "*lsxlc"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"

Call numlist.NumberListRemoveRange (2)

```

    Print "The new stream is " & numlist.Text
End Sub

```

Example Output

The new stream is 1.11, 22.2, 33, 4.444, 66, 50 - 55

NumberListRemoveValue Method for LCStream

This method removes a value at a specified position from a number list LCStream object.

The stream must be of the proper format, i.e.,
LCSTREAMFMT_NUMBER_LIST.

Defined In
LCStream

Syntax

Call *lcStream*.NumberListRemoveValue(*index*)

Parameters

index Long. The position of the value to remove.

Example

```

Option Public

Uselsx "*lsxlc"
    // DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

    Dim numlist As New LCStream (0, 0, LCSTREAMFMT_NUMBER_LIST)

    numlist.Text = "1.11, 22.2, 33, 4.444, 50-55, 66, 77-79"

    Call numlist.NumberListRemoveValue (3)

    Print "The new stream is " & numlist.Text

End Sub

```

Example Output

The new stream is 1.11, 22.2, 4.444, 66, 50 - 55, 77 - 79

TextListFetch Method for LCStream

This method fetches a text list from a text list stream object and assigns it to another stream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In
LCStream

Syntax

Call *lcStream.TextListFetch(index, format, stream)*

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| <i>index</i> | Long. The index position of the value to fetch. If the index is one, this method will accept a text stream as a single-entry text list. |
| <i>format</i> | Long. The format of the stream to return. |

Return Value

stream LCStream. The new stream object.

Example

Option Public

```
Uselstx "*lsxlc"  
    // DECS only, see overview in this chapter for LEI syntax.
```

Sub Initialize

```
Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)
```

```
Dim text As String
```

```
textlist.text = "red, orange, yellow, green, blue, indigo, violet"
```

```
Call textlist.TextListFetch (3, text)
```

```
Print "The 3rd string in the text list is " & text
```

End Sub

Example Output

```
The 3rd string in the text list is yellow
```

TextListInsert Method for LCStream

This method inserts text or a text list into a text list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In

LCStream

Syntax

Call *lcStream.TextListInsert(index, stream)*

Parameters

| <i>Value</i> | <i>Description</i> |
|--------------|--|
| index | Long. The position at which to insert the text string. |
| stream | LCStream. The text string to insert. |

Example

```
Option Public
```

```
Useslsx "*lsxlc"
```

```
    // DECS only, see overview in this chapter for LEI syntax.
```

```
Sub Initialize
```

```
    Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)
```

```
    textlist.text = "red, orange, yellow, green, blue, indigo,  
                    violet"
```

```
    Call textlist.TextListInsert (3, "black")
```

```
    Print "The new value of the text list is " & textlist.text
```

```
End Sub
```

Example Output

```
The new value of the text list is red, orange, black, yellow,  
green, blue, indigo, violet
```

TextListRemove Method for LCStream

This method removes a string from a text list LCStream object.

The stream must be of the proper format, i.e., LCSTREAMFMT_TEXT_LIST.

Defined In

LCStream

Syntax

Call *lcStream.TextListRemove(index)*

Parameters

index Long. Position of the text stream to remove.

Example

Option Public

Uselsx "**lsxlc*"

// DECS only, see overview in this chapter for LEI syntax.

Sub Initialize

Dim textlist As New LCStream (0, 0, LCSTREAMFMT_TEXT_LIST)

textlist.text = "red, orange, yellow, green, blue, indigo,
violet"

Call textlist.TextListRemove (3)

Print "The new value of the text list is " & textlist.text

End Sub

Example Output

The new value of the text list is red, orange, green, blue,
indigo, violet

Appendix A

Error Messages

This appendix provides information about error messages.

Introduction

Each entry each provides the error code in both hexadecimal and decimal format (hexadecimal is first, decimal second), the LCSTATUS constant (usable from any language), the error text for the message, and then a description of the error.

Parameterized elements of a message are indicated in all uppercase, and are replaced at error-generation time with the relevant value. Optional elements of a message are enclosed in square brackets, and are included only if the contained parameters have values provided.

Anatomy of an Error Message

This section provides information about how to interpret the LSX error messages. It describes the format of each error and what each component of the error message indicates.

Example Error Message

A sample error message is shown below:

Error: Two Fields cannot have the same name within a Form, Connector Notes, Method -Create [Metadata]- (0x80000803)

Error Message Components

The error message shown above can be broken down into the following components:

“Error: ” – This prefix is added to any error. It is not included for ‘events’ (informational messages).

“Two Fields cannot have the same name within a Form” – This is the specific error message. This can be either an LC error message, or an external error message. In this case, it is an external error message.

“Connector ‘Notes’” – This is the Connector which generated the error message. This element is included only when the error occurs inside an LCConnection method – it is skipped otherwise.

“Method-Create [Metadata]” – This is the LCConnection method in which the error occurred. It is only generated when the Connector is included in the error message. It indicates the LCConnection method generating the error. In addition, for methods which accept a parameter indicating an object type (Create, Drop, or Catalog) or action type (Action), the object or action type is included in square brackets for more information.

“(0x80000803)” – This is the external error code generated by the back-end system. It is only included when the actual error was LCFAIL_EXTERNAL – which is the case whenever the error is not an LC error. When the value is between -65536 and 65536, the decimal value is used. When it is lower or higher, the hex value is used.

Format of Errors

Errors appear in the following format:

- Hex Value
- Decimal Value
- Constant
- Text
- Description

General Errors

Two general LEI error messages are shown below:

&H101 257 LCFAIL_PROGRAM
Lotus Connector program failure – contact Lotus support

An unexpected internal error has occurred. Collect all information regarding events leading up to this error and contact Lotus technical support.

&H102 258 LCFAIL_MEMORY
Unable to allocate memory

An attempt to allocate program memory failed. There are a few possible causes of this error. First check system resources to determine if your system is low on memory. If it is, then more memory is needed on your system to support the processing tasks occurring. If your system still has significant memory available, then a request for a very large amount of memory was made. Certain operations are memory intensive, such as loading extremely

large data values, requesting too many records at one time, or using a metaconnector such as the order metaconnector (which loads the entire result set at once) against a very large result set.

Connector Errors

Several connector-related error messages are shown below:

&H3001 12289 LCFAIL_UNAVAILABLE
Requested functionality is not available

A request for unsupported functionality was made. This can occur when using a Connector which doesn't support a specific operation (for example, a Connector may not support the Create method), or when an LC API method doesn't support a specific request (see the documentation for the method in question).

&H3002 12290 LCFAIL_END_OF_DATA
The last data value has been retrieved

The last element has been retrieved from a list of available data values. This is a normal return code from any operation which is called to iterate through data values (for example, LCConnection.Fetch, LCConnect.ListProperty, LCFieldlist.List, etc...). It does not indicate a fatal error, and is usually part of a normal flow of operations.

&H3003 12291 LCFAIL_INVALID_INDEX
Cannot locate list element

A request made of a multi-value object provided an invalid index. This normally occurs when accessing a list entity (for example, a field in a fieldlist or a text entry in a text list) where the index provided is greater than the maximum valid index for that object. Note that all indices are one-based, so that an object with three elements has valid index values of 1, 2, and 3.

&H3004 12292 LCFAIL_INVALID_LIST
Invalid List direction

When using a method which iterates through values, an LCLIST parameter indicates the listing method (for example, start at the first, or return the next entry). Either an invalid listing direction was provided, or a valid direction was provided, but is not supported in the current context.

&H3005 12293 LCFAIL_INVALID_CONVERT
Invalid conversion

An unsupported data conversion was attempted. Conversion is valid between members of the same data type “class” (numbers, streams, and datetimes) as well as to and from text. Conversions between datatype classes where one type is not text (for example, converting between a datetime and a currency) will return this error. In addition, conversion to or from a binary stream formats is generally only valid when one type is text or a binary BLOB format.

&H3006 12294 LCFAIL_INVALID_TEXT_LIST
This operation requires a valid text list

An action that operates on a text list stream (a binary stream of format LCSTREAMFMT_TEXT_LIST) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted text list.

&H3007 12295 LCFAIL_INVALID_NUMBER_LIST
This operation requires a valid number list

An action that operates on a number list stream (a binary stream of format LCSTREAMFMT_NUMBER_LIST) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted number list.

&H3008 12296 LCFAIL_INVALID_DATETIME_LIST
This operation requires a valid datetime list

An action that operates on a datetime list stream (a binary stream of format LCSTREAMFMT_DATETIME_LIST) was provided a stream of a different format or with an invalid structure. Ensure that the stream data is a properly formatted datetime list.

&H3009 12297 LCFAIL_ZERO_INDEX
All index values are one-based – an index of zero is not valid

A request made of a multi-value object provided an index of zero. All objects and methods which support an index reference (for example, fieldlists and text lists) use one-based indices, zero is never a valid index value.

&H300A 12298 LCFAIL_ZERO_COUNT
This operation requires a non-zero count

A parameter indicating a count was given a value of zero when it is not valid. Zero counts are only valid in specific cases – check the method documentation for more information.

&H300B 12299 LCFAIL_ZERO_OFFSET

All offset values are one-based – an offset of zero is not valid.

A request made of a multi-value object provided an offset of zero. All methods which support an offset parameter (for example, LCStream.Merge) use one-based offsets, zero is never a valid offset value.

&H300C 12300 LCFAIL_ZERO_FORMAT

This operation requires a non-zero Stream Format

A parameter indicating a stream format was given a value of zero when it is not valid. Zero stream formats are only valid in specific cases; check the method documentation for more information. A list of valid stream formats is provided in your documentation.

&H300D 12301 LCFAIL_NULL_BUFFER

A NULL buffer was provided when one was required

A method with a required buffer parameter was provided no value where a valid buffer was required. Check the method documentation for more information.

&H300E 12302 LCFAIL_NULL_RESULT

A return parameter is required, but none was provided

A method with a required output parameter was provided no value where a valid parameter was required. Check the method documentation for more information.

&H300F 12303 LCFAIL_FIXED_LENGTH

A fixed-length stream requires a non-zero length

When requesting a fixed-length stream (with the stream flag LCSTREAMF_FIXED), a maximum length property with a non-zero value must be provided via LCStream.Create.

&H3010 12304 LCFAIL_INVALID_FLAGS

The supplied flags are invalid, possibly due to a conflict

Bitwise flags provided for an operation are not valid; either invalid flags were specified, or conflicting flags were provided. Check the object or method documentation for more information on valid flags.

&H3011 12305 LCFAIL_TEXT_TRANSLATE

Text translation failure

Translation between character sets failed. This may be caused by an incorrect character set indicator or invalid data in the source text. If the problem persists, contact Lotus technical support.

&H3012 12306 LCFAIL_NULL_FIELDNAME
A NULL field name was provided

A NULL or empty field name was provided when a valid field name was required. Ensure that the field name being provided has at least one character, and check the method documentation for more information.

&H3013 12307 LCFAIL_INVALID_FIELDLIST
Invalid fieldlist

An LCFIELDLIST object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new fieldlist. Handles are no longer valid once the object is freed.

&H3014 12308 LCFAIL_INVALID_CONNECTION
Invalid connection

An LCCONNECTION object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new connection. Handles are no longer valid once the object is freed.

&H3015 12309 LCFAIL_EMPTY_FIELDLIST
This operation cannot be performed on a fieldlist with no fields

A fieldlist that contains no fields is not valid in this context. To avoid this error, ensure that any fields are added to the fieldlist, and check the method documentation for more information.

&H3016 12310 LCFAIL_NAME_FIELDLIST
This operation cannot be performed on a name-only fieldlist

A fieldlist created with a record count of zero is not valid in this context. Such fieldlists are intended for name mapping, and cannot contain data values. As such, they cannot be used in any situation where data will be read from or written to a fieldlist. To avoid this error, provide a non-zero record count when creating the fieldlist.

&H3017 12311 LCFAIL_NATIVE_OVERRIDE
Native text format override supplied is not a valid stream format indicator

An override of the native character set was provided in an .INI variable as indicated in the documentation. The character set indicated is not a valid character set identifier. Check the documentation for a list of valid character set strings.

&H3018 12312 LCFAIL_RECORD_INDEX

An invalid fieldlist record index was encountered

A fieldlist record index is not valid for the relevant fieldlist. This generally occurs with LCConnection class methods which accept a fieldlist and a record index. The index must be at least one (all indices are one-based), and cannot be greater than the number of records that the fieldlist was created with.

&H3019 12313 LCFAIL_RECORD_COUNT

Request to transfer more records than allocated in fieldlist

A fieldlist record count is not valid for the relevant fieldlist. This generally occurs with LCConnection class methods which accept a fieldlist and a record count. The count plus the record index cannot exceed the number of records that the fieldlist was created with. In addition, a count of zero is only valid in specific situations. See the method documentation for more information.

&H301A 12314 LCFAIL_LIST_SETUP

Fieldlist iteration requires initial setup

Listing fieldlist in a fieldlist with LCFieldlist.List requires that LCFieldlist.ListSetup be called first to prepare the iteration.

&H301B 12315 LCFAIL_NO_MERGE_DATA

The data fieldlist in a merge cannot be a name-only fieldlist

When using the methods LCFieldlist.Merge or MergeVirtual, the fieldlist provided as the data fieldlist cannot be a name-only fieldlist. It must be created with a non-zero record count.

&H301C 12316 LCFAIL_NO_RESULTSET

This operation requires an active result set

A requested LCConnection operation cannot be performed unless the connection has an active result set. Produce a result set before attempting this operation. This error is generally produced by calling LCConnection.Fetch without a previous call to a method which produces a result set (such as Execute, Select, Call, or Catalog).

&H301D 12317 LCFAIL_NO_WRITEBACK

This operation requires an active writeback result set

A requested LCCONNECTION operation cannot be performed unless the connection has an active writeback result set. Produce a writeback-enabled result set before attempting this operation. This error is generally produced by calling LCCONNECTION.UPDATE or REMOVE with the "Writeback" property set, but without a previous call to a method which produces a writeback result set (such as EXECUTE or SELECT with the "Writeback" property set). If the result set is produced without the writeback property set, then writeback operations are not supported against that result set.

&H301E 12318 LCFAIL_WRITEBACK_COUNT

Writeback operation record counts must be one

When requesting a writeback operation against an LCCONNECTION, only the most recently fetched record is affected, so the record count must be one.

&H301F 12319 LCFAIL_TRANSLATE_INIT

Text translation subsystem initialization failure

The internal character set translation subsystem failed to initialize properly. This is generally due to an improperly installed environment. Check the document for installation and platform-specific notes, and contact Lotus technical support if the problem persists.

&H3020 12320 LCFAIL_TIMEBOMB

This time-limited version has expired [as of DATE]

This is a time-limited version which expired on the date indicated. A new version with a later timebomb, or a complete version must be obtained from Lotus.

&H3021 12321 LCFAIL_SESSION_NOT_INIT

The Lotus Connector Session must be initialized before performing any operation

The Lotus Connector Session class must be initialized before any other LC objects can be created or methods used.

&H3022 12322 LCFAIL_CONNECTOR_VERSION

Incorrect Connector version

A Connector was built with an incompatible version of the Lotus Connector Toolkit. Contact Lotus or the developer of the Connector for an updated version.

&H3023 12323 LCFAIL_NOT_CONNECTED

This operation requires a connection to a Connector

The LCCConnection method called cannot be used on an unconnected LCCConnection. Call the Connect method before this operation. In general, any operation which produces a result set or manipulates data or metadata cannot be called before connecting – only getting and setting properties may occur. One exception for some Connectors is that a server or database catalog result set may be produced and fetched from before connecting.

&H3024 12324 LCFAIL_CONNECTED

This operation cannot be performed when there is a valid connection to a Connector

The LCCConnection operation performed is not valid when the LCCConnection is actively connected – the Disconnect method must first be called. This may occur when performing an action which affects the connection itself, such as using the Drop method on a database.

&H3025 12325 LCFAIL_EXTERNAL

<external error text>

An external database or application accessed by a Connector generated an error. The external error text and code are available with the method LCSession.GetStatus.

&H3026 12326 LCFAIL_ACTIVE_SUBCONNECTOR

The sub-Connector of a metaconnector can only be set once

Any subconnector property of a metaconnector can only be set once; any attempt to reset it will generate this error.

&H3027 12327 LCFAIL_TRANSLATE_TABLE

No translation tables are available for the character set

A translation table is available for most translations between supported character sets. This error indicates that the translation table for a specific character set or translation is not available and the character set translation cannot be performed. Often, a similar character set is available and can be used to obtain the same effective translation.

&H3028 12328 LCFAIL_NO_SCROLL

This operation requires an active scrollable result set

Using the LCCConnection.Fetch method with a negative record count is only valid when the Connector supports scrolling result sets and the active result set was produced with the “Scrolling” property set. This error is returned from the Fetch method when a negative record count is provided and either of these conditions is not met.

&H3029 12329 LCFAIL_BINARY_FORMAT

This operation requires a non-binary Stream Format

Certain LCStream methods are not valid on formatted binary stream formats. This method requires a stream with a text format, or unformatted BLOB binary format. Check the method documentation for more information.

&H302A 12330 LCFAIL_ASYNC_ACTIVE

Asynchronous operation is still active

This error is returned from methods that wait for an asynchronous operation to complete. When the asynchronous operation is still active and any wait period specified passes, this error is returned.

Parameterized Connector Errors

Several parameterized connector errors are shown below:

&H3101 12545 LCFAIL_INVALID_METADATA

Metadata object ['METADATA'] does not exist

The value provided for the “Metadata” property is not a valid metadata object in the external system. The invalid metadata name will generally be provided. Ensure that a valid value is being provided; common errors include misspellings, incorrect capitalization in case-sensitive systems, and being connected to the wrong database.

&H3102 12546 LCFAIL_TYPE_MISMATCH

Type mismatch [for field 'FIELDNAME'] [; Connector: TYPE1] [, External: TYPE2]

Data type mismatch. Implicit data conversion is supported only within a datatype “class”: number, stream, or datetime. When mapping is attempted between datatypes in different classes (for example, INT to TEXT; or DATETIME to NUMERIC), this error is generated. The fieldname and datatypes are generally provided in the error text. Ensure that the datatypes being mapped are members of the same datatype class.

&H3103 12547 LCFAIL_DUPLICATE

Duplicate object ['NAME']

An object of the same name already exists; select a different name. This error can occur when creating an API object or an external system object, such as creating a new metadata object where one of the same name already exists.

&H3104 12548 LCFAIL_FIELD_COUNT_MISMATCH
Field count mismatch [; Connector: COUNT1, External: COUNT2]

The number of fields did not match between a Connector and an external system. In some cases, this problem can be resolved automatically (for example, when writing fields to a back-end system that supports more fields than provided). In other cases, this error will be generated (for example, when writing 5 fields to a back-end system which supports 3 fields). To avoid this error, fields can be removed from a fieldlist on a per-operation basis by using field flags.

&H3105 12549 LCFAIL_KEY_COUNT_MISMATCH
Key count mismatch [; Connector: COUNT1, External: COUNT2]

The number of keys did not match two systems. In cases where keys must be specified for two systems, the same number of keys must be provided for both systems.

&H3106 12550 LCFAIL_STAMPFIELD_TYPE
Timestamp field ['FIELDNAME'] must be type Datetime [; Actual: TYPE]

A timestamp field provided for the “StampField” property of a Connector must represent an external field of type datetime. This error is returned when it is a number or stream datatype.

&H3107 12551 LCFAIL_FIELD_TYPE
Type mismatch for field ['FIELDNAME'] used in this context [; Expected: TYPE1 [, Actual: TYPE2]]

Specific functionality requested of a Connector requires that a field be of a particular datatype. This error is generated when a specific datatype is expected for a particular field, but an incompatible datatype is provided.

&H3108 12552 LCFAIL_MERGE_FIELD
Field mapping failed due to a missing field ['FIELDNAME']

When using the methods LCFieldlist.Merge or MergeVirtual, there was a mismatch between the fieldlists. Either a field in the namelist had no match in the data list, or vice-versa. If such a mismatch occurs and the merge flag LCMERGEF_NAME_LOSS or LCMERGEF_DATA_LOSS is not specified to indicate that loss of fields is allowed, then this error is generated. Ensure that the field indicated in the error has a corresponding field in the other fieldlist.

&H3109 12553 LCFAIL_MISSING_PROPERTY
No value supplied for required property ['PROPERTY']

The indicated property requires a value for the requested operation, but none was provided. Supply a valid value for the property. Check the documentation for the Connector or the method being called (for example, when using the LCConnection.Create method to create an index, the property “Index” is required, and if no value was given, this error would be generated).

&H310A 12554 LCFAIL_PROPERTY_CONFLICT
Conflicting values for properties ['PROPERTY1' and 'PROPERTY2']

The values provided for the two indicated properties conflict with one another. Check the Connector documentation for information regarding conflicting property values for this Connector.

&H310B 12555 LCFAIL_INVALID_PROPERTY
Invalid property ['PROPERTY']

The attempt to get or set the indicated property failed because it is not supported by the Connector. Check the Connector documentation for more information on supported properties for this Connector.

&H310C 12556 LCFAIL_PROPERTY_VALUE
Invalid property value [for property 'PROPERTY']

The value provided for the indicated property is not a valid value. Check the Connector documentation for information on valid property values for this Connector.

&H310D 12557 LCFAIL_INVALID_CHARSET
The text format provided ['CHARSET'] is not a valid Lotus Connector character set indicator

An attempt to indicate a character set by its textual representation failed due to an invalid string. Check the documentation for a list of supported character set strings.

&H310E 12558 LCFAIL_READ_ONLY_PROPERTY
Cannot change the value of read-only property ['PROPERTY']

The indicated property is read-only for this Connector and an attempt to set its value was made. For example, the “TextFormat” property of some Connectors cannot be assigned, so will generate this error on any such attempt.

&H310F 12559 LCFAIL_MISSING_CONNECTORLCX
Cannot load Connector LCX Library ['CONNECTOR']

The Connector library .LCX file could not be loaded. Ensure that a valid Connector was specified and that the Connector is available on the system. A common mistake which produces this error is to not have any client files installed to support the Connector (for example, use of the DB2 Connector requires that a DB2 client product be installed).

&H3110 12560 LCFAIL_INVALID_CONNECTORLCX
Invalid Connector LCX Library ['CONNECTOR']

The Connector library .LCX file was located and loaded, but was not a valid Connector. Contact Lotus or the Connector vendor for a valid Connector library.

Field-Specific Parameterized Connector Errors

Several field-specific parameterized connector errors are shown below:

&H3201 12801 LCFAIL_OVERFLOW
Data overflow[in field 'FIELDNAME']

A data overflow was encountered when transferring data to or from a field. This error generally occurs when writing data to an external system, and the data is too large for the target field (for example, writing the text “abcdefg” to a SQL CHAR(5) field). This error can also occur, though, when reading data into an internal datatype and the data cannot be accommodated (a number or datetime is out of range, or a stream exceeds the maximum allowed length. This error can be suppressed by assigning the field flag LCFIELDF_TRUNC_DATA, or selecting the corresponding UI option to allow data loss on overflow.

&H3202 12802 LCFAIL_PRECISION_LOSS
Data precision loss [in field 'FIELDNAME']

This error is generated when performing type mapping between internal and external fields when the datatypes are different, and precision may be lost on data transfer. Note that this error indicates that precision will be lost between the datatypes, irrelevant of the actual data (for example, writing a FLOAT values of 3 to an INT field will be considered a precision loss, since it checks the datatypes once at mapping time, rather than the data values for each transfer). This error can be suppressed by assigning the field flag LCFIELDF_TRUNC_PREC, or selecting the corresponding UI option to allow precision loss.

&H3203 12803 LCFAIL_INVALID_INT
Invalid integer value [in field 'FIELDNAME']

The indicated field contains an invalid integer value, or an attempt to assign an invalid integer value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3204 12804 LCFAIL_INVALID_FLOAT
Invalid float value [in field 'FIELDNAME']

The indicated field contains an invalid floating point value, or an attempt to assign an invalid floating point value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3205 12805 LCFAIL_INVALID_CURRENCY
Invalid currency value [in field 'FIELDNAME']

The indicated field contains an invalid currency value, or an attempt to assign an invalid currency value was made. Ensure that the data source for the field was a valid datetime value within the allowable range of an LCCurrency and check that the field value is not being destroyed by an incorrect operation or method call.

&H3206 12806 LCFAIL_INVALID_NUMERIC
Invalid numeric value [in field 'FIELDNAME']

The indicated field contains an invalid numeric value, or an attempt to assign an invalid numeric value was made. Ensure that the data source for the field was a valid numeric value within the allowable range of an LCNumeric and check that the field value is not being destroyed by an incorrect operation or method call.

&H3207 12807 LCFAIL_INVALID_DATETIME
Invalid datetime value [in field 'FIELDNAME']

The indicated field contains an invalid datetime value, or an attempt to assign an invalid datetime value was made. Ensure that the data source for the field was a valid datetime value within the allowable range of an LCDatetime, and check that the field value is not being destroyed by an incorrect operation or method call.

&H3208 12808 LCFAIL_INVALID_STREAM
Invalid stream value [in field 'FIELDNAME']

The indicated field contains an invalid stream value, or an attempt to assign an invalid stream value was made. Ensure that the data source for the field was a valid value and check that the field value is not being destroyed by an incorrect operation or method call.

&H3209 12809 LCFAIL_INVALID_FIELD
Invalid field ['FIELDNAME']

An LCFIELD object handle is either not a valid handle, or is zero. Object handles must be a valid value returned by a method which creates a new field. Handles are no longer valid once the object is freed.

&H320A 12810 LCFAIL_INVALID_TYPE
Invalid data type[for field 'FIELDNAME']

The datatype for the indicated field is an overview datatype, or a datatype parameter was provided with an invalid value. This error will generally only occur if some sort of corruption exists, but could also be indicative of a version mismatch between a specific Connector and the calling program or tool. Check to ensure that only valid datatypes of the form LCTYPE_XXX are being used, and if the problem persists, contact Lotus technical support.

&H320B 12811 LCFAIL_INVALID_KEY
Invalid key field ['FIELDNAME']

The indicated field was provided as a key field, but no such searchable field exists in the result set or external system. Two common sources of this error are misspellings of the key field name or use of a non-searchable field (as defined by the external systems – check the Connector documentation for information on datatypes that cannot be used as keys) as a key. This error can also occur when the key field is left out of the result set.

&H320C 12812 LCFAIL_DUPLICATE_KEY
Duplicate key field ['FIELDNAME']

The indicated field was provided as a key field twice; a single field can only be used once as a key. Either remove the second occurrence of the key field, or provide a different field name in its place.

&H320D 12813 LCFAIL_INVALID_STAMPFIELD
Invalid timestamp field ['FIELDNAME']

The indicated field was provided as a timestamp field, but no such field exists in the result set or external system. This generally occurs because the name of the field was misspelled, or it was not included in the result set.

&H320E 12814 LCFAIL_INVALID_FIELDNAME
Field name ['FIELDNAME'] is not valid in this context

The indicated field name is not valid for the context in which it was used. For example, when attempting to create a new metadata object in an external system, this error may be generated if any of the fields are not valid field names for that external system.

&H320F 12815 LCFAIL_VIRTUAL_FIELD
Unsupported virtual field ['FIELDNAME']

Virtual fields are an advanced feature only supported by certain Connectors. When using virtual fields, only those fields supported by the Connector in question are valid. When a Connector supports virtual fields and a field provided has that Connector's virtual code set, then it will generate this error if it does not understand the supplied virtual field. Check the Connector documentation for information on supported virtual fields.

&H3210 12816 LCFAIL_VIRTUAL_VALUE
Invalid data value for virtual field ['FIELDNAME']

Virtual fields are an advanced feature supported only by certain Connectors. When using virtual fields, only values supported by the Connector in questions are valid. When a Connector supports virtual fields and a field provided has that Connector's virtual code set, then it will generate this error if it cannot handle or interpret the value supplied in that virtual field. Check the Connector documentation for information on supported virtual fields and valid values for those fields.

&H3211 12817 LCFAIL_INVALID_ORDER
Invalid order field ['FIELDNAME']

The indicated field was provided as an ordering field, but no such usable field exists in the result set or external system. This error can occur when misspelling a fieldname or leaving the indicated field out of the result set. It can also occur in some cases when using a non-searchable field (as defined by the external systems; check the Connector documentation for information on datatypes that cannot be used as keys) for ordering, and not using an Order metaconnector.

Appendix B

Connector Property Tokens

This appendix provides information about the property tokens used in the Lotus Connectors LotusScript Extensions.

- Flags can be combined with LCX property tokens.
- The base value for an LCX property is between 1 and 0xFFFFF.

Predefined Tokens

Several connector property token are shown below:

LCTOKEN_NAME – Library name for the connector used to create this connection.

LCTOKEN_CONNECTOR_CODE – A code unique to this connector, and the same for all connections to this connector. This code is assigned dynamically, so may change for each execution of a script. This code can be used as a virtual field code to indicate special handling by any connection to this connector.

LCTOKEN_CONNECTION_CODE – A code unique to this connection among all connections. This code is assigned dynamically, so may change for each execution of a script. This code can be used as a virtual field code to indicate special handling by this connection only.

LCTOKEN_EVENT_ERROR – Error code (LCFAIL_XXX constant) to treat as an event. When this error is generated, it is downgraded to an informational event. Set to LCSUCCESS to clear this property.

LCTOKEN_CHARACTER_SET – Character Set indicator representing the character set used by this connector for data transfers to and from the back-end system. This is normally dynamically determined by the connector from the back-end system at run time. Often read-only, but may be assigned for some connectors. Valid values include any text stream format suffix from the supported character set appendix (e.g., "IBMCP932" for LCSTREAMFMT_IBMCP932.)

LCTOKEN_IGNORE_ERROR – Error code (LCFAIL_XXX constant) to ignore. When this error is generated, it is ignored. Set to LCSUCCESS to clear this property.

LCTOKEN_LCX_VERSION – Version code for the connector LCX library.

LCTOKEN_CONNECTOR_NAME – Property representing the first sub-connector for a particular Metaconnector. Setting this property to a connector name is the same as creating a connection of that connector type and assigning it to the metaconnector.

LCTOKEN_SERVER – Server string, used for connectivity.

LCTOKEN_DATABASE – Database string, used for connectivity.

LCTOKEN_USERID – User ID string, used for connectivity.

LCTOKEN_PASSWORD – Password string, used for connectivity. This value is write-only, and cannot be retrieved from a connector once assigned.

LCTOKEN_METADATA – Metadata object name. Used for reading and writing records, as well as creating, dropping, and cataloging metadata and fields.

LCTOKEN_INDEX – Index string. Used when an index name is required, such as when creating or dropping an index.

LCTOKEN_MAP_NAME – Map by name flag; set to TRUE to map fields in fieldlist to back-end metadata by name, and to FALSE to map by position.

LCTOKEN_WRITEBACK – Writeback flag; set to TRUE before calling Select to produce a writeback result set, and before calling Update or Remove to perform a writeback operation against a writeback result set.

LCTOKEN_FIELDNAMES – Selection field name list, a text list (or comma or semicolon separate list of field names in a text string) indicating fields to include in a Select method result set. By default, all fields selected are included; this property can be used to restrict that list.

LCTOKEN_ORDERNAMES – Ordering field name list, a text list (or comma or semicolon separate list of field names in a text string) indicating fields to order a Select method result set by.

LCTOKEN_CONDITION – Condition string in a syntax defined by the connector. This clause will be embedded within a selection or modification statement for Select, keyed Update, or keyed Remove operations.

LCTOKEN_STAMPFIELD – Name of the field containing a timestamp value (must be a datetime datatype). When set before calling the Select method, the result set will be restricted to fields where the timestamp field value is between the value assigned to the BASESTAMP property and returned in the MAXSTAMP property. If BASESTAMP is not set, no minimum will be used. The Select method will determine the current time from the back-end server, assign that value to the MAXSTAMP property, and use that datetime as the upper boundary for selection.

LCTOKEN_BASESTAMP – Base timestamp value for timestamp selection; see STAMPFIELD property.

LCTOKEN_MAXSTAMP – Maximum timestamp value for timestamp selection; see the STAMPFIELD property. This property is set by the Connector during Select operations when the STAMPFIELD property is set.

LCTOKEN_TEXT_FORMAT – See CHARACTER_SET property; this property is the numeric constant assigned to the particular character set.

LCTOKEN_PROCEDURE – Procedure name to execute for the Call method.

LCTOKEN_OWNER – Owner name string to restrict Catalog method calls (excluding server and database catalogs) to only objects owned by this owner.

LCTOKEN_SCROLLABLE – Whether to produce a scrollable result set. This property is not supported by any connectors at this time.

Connector Identification Property Tokens

The connector identification property tokens are listed below:

LCTOKEN_IDFLAG_ACTION – Action flags

LCTOKEN_IDFLAG_CONNECTOR – General flags

LCTOKEN_IDFLAG_OBJECT_CATALOG – Catalog flags

LCTOKEN_IDFLAG_OBJECT_CREATE – Create flags

LCTOKEN_IDFLAG_OBJECT_DROP – Drop flags

LCTOKEN_IDNAME_SERVER – Name for server objects in this back-end system.

LCTOKEN_IDNAME_DATABASE – Name for database objects in this back-end system.

LCTOKEN_IDNAME_USERID – Name for user ID objects in this back-end system.

LCTOKEN_IDNAME_PASSWORD – Name for password objects in this back-end system.

LCTOKEN_IDNAME_METADATA – Name for metadata objects in this back-end system.

LCTOKEN_IDNAME_FIELD – Name for metadata fields in this back-end system.

LCTOKEN_IDNAME_ALT_METADATA – Name for alternate metadata objects in this back-end system.

LCTOKEN_IDNAME_ALT_FIELD – Name for alternate metadata fields in this back-end.

LCTOKEN_IDNAME_PROCEDURE – Name for procedure objects in this back-end system.

LCTOKEN_IDNAME_INDEX – Name for index objects in this back-end system.

LCTOKEN_IDNAME_PARAMETER – Name for procedure parameters in this back-end system.

Appendix C

Connector Properties

This chapter provides information about Connector properties. These properties are used when creating Connections using the Lotus Connectors LotusScript Extensions.

Connector Properties

The table below describes the items included in the tables of Connector properties.

| <i>Item</i> | <i>Description</i> |
|-------------|---|
| Token | Property token. Indicates the property is represented by the pre-defined constant LCTOKEN_<name>. For example, METADATA indicates the constant LCTOKEN_METADATA. A number indicates a property specific to the particular connector. |
| Name | The property name. This is the dynamic property added through the LSX. |
| Type | The data type of the property: Boolean Considered either FALSE (zero) or TRUE (non-zero). Integer Valid values are indicated in the description. Datetime Standard datetime. Text Standard text stream. Text List Can be submitted as a formatted text list or (more commonly) a comma, semicolon, or newline-separated text value. All default values are FALSE (boolean), 0 (integer), no value (datetime), and the empty string (text) unless otherwise indicated. |

Notes Connector Properties

The table below defines the properties for the Lotus Connector for Notes.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|--|-------------|
| LCTOKEN_SERVER | Server Notes Server. If not supplied, the local Notes database will be accessed. | Text |
| LCTOKEN_DATABASE | Database Notes database filepath, not the title. Required. Primarily used for establishing a connection. | Text |
| LCTOKEN_METADATA | Metadata Notes form name. Unlike many other connectors, Notes Execute operations also require a Metadata property value, since the selection formula does not include metadata information. | Text |
| LCTOKEN_INDEX | Index Notes view name. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Text list of fields to select - used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Text list of fields to order results by - used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition Notes-specific syntax conditional clause used for Select operations (part of a Notes selection formula). This must be valid syntax for a “select <condition>” formula. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|---|-------------|
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field - used for Select operations. Using StampField without a view (DB search instead) only works when the timestamp field is “@Modified“, and the LoadModified property is set to TRUE. When not using this new property, a view search is still required (although if no view property is set, a temporary one will be created). | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results - used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operation, set by the connector to the current/maximum timestamp value - set by Select operations. | Datetime |
| 1 | Port Port name to locate the server on. No value searches all ports. | Text |
| 2 | EnforceForm Whether to enforce the form design on documents created and accessed. 0 — The connection does not execute any Notes form- and field-related formulas. 1 — Executes all Notes field-related formulas (default value, input translation, and computed field) as part of the session. This option may cause slower data transfer as a result of formula calculations. This option does not execute @DB functions. 2 — The session sets all field flags as defined in the form but does not execute any formulas. As a result, special types, indicated in Notes by flags --such as reader and author names--are assigned in new documents as indicated in the form. This option avoids the overhead of computing field formulas. | Integer |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 3 | MultiValueAsText Whether to represent all multi-value Notes types (text list, number list, and datetime list) as text values. This can make type transformation between non-Notes connectors that do not support multi-value types more straightforward. | Boolean |
| 4 | BulkStore Whether to buffer all document modifications for bulk submittal at disconnection. This increases performance, but prevents access to those changes until disconnection. | Boolean |
| 5 | FilePath Directory on the Domino Server into which attachments are extracted to and attached from when using the FILE virtual field or the LoadFile property. By default, the current working directory is used. | Text |
| 6 | UpdateViews Whether to update all views in the database during disconnection. After making extensive changes, the initial opening of a view can perform a time-consuming update. | Boolean |
| 7 | CreateDatabase Whether to create a database during connection if one does not exist. The database is blank, unless a database template is specified with the TemplateServer and TemplateDatabase properties. | Boolean |
| 8 | TemplateServer Notes Server where the database specified at TemplateDatabase resides. This template is used in conjunction with the CreateDatabase option. No value uses the local Notes installation | Text |
| 9 | TemplateDatabase Notes template database filepath on TemplateServer. This template is used in conjunction with the CreateDatabase option. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 10 | View Notes view to open for Execute or to use for keyed operations. The view's result set replaces any Statement submitted to Execute. | Text |
| 11 | Agent Notes agent to run for an Execute operation on the result set produced by either the Statement parameter or the View property. | Text |
| 12 | FullTextQuery Notes full text query for an Execute operation on the result set produced by either the Statement parameter or the View property, and any Agent property. | Text |
| 13 | TextMaxLength Notes text data is limited to slightly less than 64K. To simplify data creation on other connectors with shorter text columns, specify a value to assign as the maximum length for text metadata in any result set produced. | Integer |
| 14 | FetchViewData Whether to fetch data from view columns rather than from document fields. Requires a View property value. | Boolean |
| 15 | AllForms Whether to include documents of all forms. By default, only documents of the form indicated by the metadata property value are included. When including documents of other forms, fields in the indicated metadata are used in place of those document's forms. | Boolean |
| 16 | ViewResponses Whether to include view response documents. By default, only main topics are included. Requires a View property value. | Boolean |
| 17 | LoadUnid Whether to add a UNID virtual field to the result set produced by an Execute or Select operation. | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|--|-------------|
| 18 | LoadNoteid Whether to add a NOTEID virtual field to the result set produced by an Execute or Select operation. | Boolean |
| 19 | LoadRef Whether to add a REF virtual field to the result set produced by an Execute or Select operation. | Boolean |
| 20 | LoadFile Whether to add a FILE virtual field to the result set produced by an Execute or Select operation. | Boolean |
| 21 | CopyHierarchy Whether to add a LCXHIER virtual field to the result set produced by an Execute operation. | Boolean |
| 22 | CopyFile Whether to add a LCXFILE virtual field to the result set produced by an Execute or Select operation. | Boolean |
| 23 | ExecuteDelete Whether to delete the result set produced during an Execute operation. This property is to simulate native DELETE statements provided on some other connectors. | Boolean |
| 24 | InsertMail Whether to reroute Inserted documents as mail. Normal Notes addressing information, such as the required SendTo field, is used. | Boolean |
| 25 | MailEmbedForm Whether to embed the form in documents mailed through the InsertMail property. Use when the target form is not available in the recipients' mail database. | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 26 | <p>Encrypt</p> <p>Encrypts all encryption-enabled fields in the form of the target document. To use private encryption keys (the default is to use the public key), provide one or more private encryption key names, separated by commas or semicolons in EncryptKeyList. These keys must be available on the Domino Server.</p> | Boolean |
| 27 | <p>EncryptKeyList</p> <p>One or more private encryption key names separated by commas or semicolons, to be used by Encrypt. Valid only if Encrypt is True.</p> | Text list |
| 28 | <p>PurgeDeleteStubs</p> <p>Clears all deletion stubs from the Notes database when disconnecting. It is strongly recommended that this only be used with UpdateViews, since once the stubs are deleted, they will not be properly removed from views.</p> | Boolean |
| 29 | <p>SearchNoCase</p> <p>Makes view searches case-insensitive (they are case-sensitive by default).</p> | Boolean |
| 30 | <p>AlterView</p> <p>When the LSX does a Select operation with the connector's View property set, AlterView=True allows the script to modify or create the view if it doesn't already exist or doesn't have the proper formatting. Necessary for select operations when a view and stampfield are specified.</p> | Boolean |
| 31 | <p>LoadModified</p> <p>Set to add a field “@Modified” to the result set, which contains the Notes implicit document timestamp. This field can only be read and attempts to set it will be ignored.</p> | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 32 | StampViewKey Indicates the value for the first key value for keyed operations when the view is an LSX-created view for timestamp selection. Zero means don't use. 1 is the value for docs outside the timestamp selection range, 2 is the value for docs inside the range. When performing a timestamp selection, this is automatically set to 1 by the connector, and reset to zero when clearing the result set, and should only be used or altered with great care. | Integer |
| 33 | CopyCompSpecial Copies any composite support fields associated with transferred documents. This includes the fields, links, and fonts to return full fidelity of composed data. To use this option from an Activity with manual field mapping, add a field 'LCXHIER' to both field lists (this is not necessary for automatic field mapping by name or position). | Boolean |
| 34 | DeleteDatabase Deletes the database specified in the Activity upon connection. This is used together with Creation (below) to delete and then recreate the database prior to population. | Boolean |
| 35 | PostFetchFormula Enter the Notes formula that will execute just after the data is fetched. The data is altered as part of the fetch/select operation. | Text |
| 36 | PreInsertFormula Enter the Notes formula that will execute just before the selected data is inserted into the target. The data is changed as part of the insert operation. | Text |
| 37 | PreUpdateFormula Enter the Notes formula that will execute before the data in the designated target is updated. The transformation will take place as part of the update operation. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 38 | AddSubformList To support computed subforms, list all of the subforms to include here. All fields in these subforms will be considered to be part of the form. | Text List |

Notes Connector Property Combinations

The following are *invalid* combinations:

- FetchViewData TRUE and Agent or FieldList non-NULL
- View NULL and any of FetchViewData, ViewResponses, or CopyFile TRUE
- ViewResponses TRUE and CopyHierarchy TRUE
- FetchViewData TRUE and any of LoadUnid, LoadNoteid, LoadRef, LoadFile, or CopyFile TRUE
- MapByName TRUE and any of LoadUnid, LoadNoteid, LoadRef, LoadFile, CopyFile, or CopyHierarchy TRUE

When using ordering or timestamps (properties OrderNames or StampField, respectively) in Notes select operations, the View property can be set to indicate the name of the view to use for these operations. When performing selections with View set, the LSX will check that the view has the proper formatting (special requirements for either ordered or timestamped result sets). If the view has the correct format, it will be used. Supplying a view name when doing these operations can drastically improve performance by not requiring the database to be reindexed every run. When no view is specified, a temporary view is still created. When the view is the wrong format or doesn't exist, the use of the AlterView property allows the LSX to create or overwrite that view to be in its own format.

Processing Attachments Stored in RTF Fields

Notes is primarily a document management system – not a relational database management system (RDBMS). Because of this, Notes performs special handling for attachments stored in Rich Text fields. It stores the attachment data separately from the Rich Text field data. Currently you can only transfer attachments from Notes to Notes; Rich Text fields with attachments do not store correctly in the target RDBMS's such as DB2 and Oracle.

To work around this constraint, you can use the Notes Connector Extract File Attachments option to save attachments to files on disk. You can then use LotusScript to read the files, store their contents in LCFields, and save the

LCFields to binary fields in a RDBMS. This level of scripting requires LotusScript expertise.

Notes Virtual Fields

Notes recognizes a set of virtual fields – fields which are interpreted differently by the Notes connector than by other connectors. To use these virtual fields, either set the virtual code of the relevant fields to the Notes connector virtual code (obtain the property LCTOKEN_CONNECTOR_CODE from a Notes connection, and set that code into the properly names field using the LCField.SetVirtualCode method), or use one of the Notes properties which adds a virtual field to the result set. The virtual fields supported by Notes are described below:

| <i>Field Name</i> | <i>Corresponding Property</i> | <i>Data type</i> | <i>Stream Length (in bytes)</i> |
|-------------------|-------------------------------|--------------------|---------------------------------|
| UNID | LoadUnid | Binary (BLOB) | 16 |
| NOTEID | LoadNoteid | Integer | N/A |
| REF | LoadRef | Binary (BLOB) | 16 |
| FILE | LoadFile | Binary (Text List) | variable (up to 64K) |
| MODIFIED | LoadModified | Datetime | N/A |
| LCXHIER | CopyHierarchy | Binary (BLOB) | 8 |
| LCXFILE | CopyFile | Binary (BLOB) | 4 |
| LCXSPEC | CopyCompSpecial | Binary (BLOB) | 4 |

Descriptions may vary based on whether a document is being read (Fetch) or written (Insert or Update):

UNID

Read – Read the UNID of the document

Write – Set the document's UNID to the field value

NOTEID

Read – Read the NOTEID of the document

Write - ignored

REF

Read – Read the parent UNID of a response document

Write – Make the document a response to the document whose UNID is in this field

FILE

Read – Retrieve all attachments from the document, writing the files to disk. This field is set to a text list containing the filenames of all the files from the document

Write – Store each entry in the text list value as a file attachment to the document.

MODIFIED

Read – Read the last modified timestamp from the document

Write – ignored

LCXHIER

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer the entire response hierarchy beneath the document into the target database. Note that the source result set must still be valid at the time the document is written.

LCXFILE

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer all file attachments directly from the source document to the target document. Since they are transferred directly without temporary storage on disk, this is more efficient than the FILE property for Notes to Notes transfers. The source document must be the most recently fetched document in the result set at the time the document is written.

LCXSPEC

This field is only relevant when the document is both read from and written to a Notes connector. When written, the field contents stored during the read are used to access the source database and transfer specific document fields required to retain full document fidelity for doclinks and rich text fonts. Specifically the fields “\$Links” and “\$Fonts” are copied. The source document must be the most recently fetched document in the result set at the time the document is written.

DB2 Connector Properties

The table below defines the properties for the Lotus Connector for DB2.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|--|-------------|
| LCTOKEN_DATABASE | Database DB2 Database name. Required. Primarily used for establishing a connection. | Text |
| LCTOKEN_USERID | Userid User ID / name: used for logging in during connection | Text |
| LCTOKEN_PASSWORD | Password Password: used for logging in during connection | Text |
| LCTOKEN_METADATA | Metadata Data table or view: the source or target for data transfer operations. | Text |
| LCTOKEN_INDEX | Index Index name. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name (TRUE) or position (FALSE) when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Text list of fields to select – used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Text list of fields to order results by - used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition DB2-specific syntax conditional clause used for Select, keyed update and remove operations. Must be of valid syntax, as in “select * from table where <condition>”. | Text |
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field - used for Select operations | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------|--|-------------|
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value - set by Select operations | Datetime |
| LCTOKEN_PROCEDURE | Procedure The stored procedure to execute for the Call method. | Text |
| LCTOKEN_OWNER | Owner Only objects owned by the owner name in this property are included in the result set | Text |
| LCTOKEN_ALT_METADATA | Affects the behavior of METADATA and FIELD catalog types, returning a list of objects of the alternate metadata type instead of the normal metadata type. | Boolean |
| LCTOKEN_RECORD_LIMIT | Result sets produced should be limited to this many records. A value of zero (the default) indicates no limit. | Integer |
| 1 | CommitFrequency Number of data modification actions between commits. A value of zero causes a commit at disconnect; a value of one auto-commits after every action; any other value commits after that many data modification actions. | Integer |
| 2 | RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state. | Boolean |
| 3 | CreateMaxLogged Use when creating tables. The maximum length of a CLOB/BLOB column to create with logging. If the length during metadata creation is longer than this value, the options NOT LOGGED COMPACT are used. A value of zero indicates no maximum, although unbounded columns are always created with this option. | Integer |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 4 | NoJournal Whether the database contains non-journaled data. If so, the transaction isolation level is set to uncommitted read to permit SQL operations. | Boolean |
| 5 | CreateInDatabase Use to add "IN DATABASE <dbname>" to a CREATE TABLE query constructed for table creation. | Text |
| 6 | TraceSQL Include all SQL statements generated during the execution of the Activity in the Activity's log | Boolean |
| 7 | TimestampTable Query separate table for the current timestamp from DB2. Useful for Replication Activities, If DB2 is on an AS/400 or an S/390 system, the query is run against the metadata specified by the Activity. If the table is large, this query can be inefficient. You can enter an alternate table name, which is smaller in size, to query for the timestamp. The table can be of any format and should contain as few rows as possible. This approach can improve the efficiency and speed of the timestamp query. | Text |

EDA/SQL Connector Properties

The table below defines the properties for the Lotus Connector for EDA/SQL.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------------------|--|-------------|
| LCTOKEN_SERVER | Server EDA/SQL Server name. Required. Primarily used for establishing a connection. | Text |
| LCTOKEN_DATABASE | Database EDA/SQL Engine name. Used to indicate a relational database engine to use. | Text |
| LCTOKEN_USERID | Userid User ID / name: used for logging in during connection | Text |
| LCTOKEN_PASSWORD | Password Password: used for logging in during connection | Text |
| LCTOKEN_METADATA | Metadata Data table: the source or target for data transfer operations. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Multi-value text list of fields to select – used for Select operations | Text List |
| ORDER_LIST LCTOKEN_ORDERNAMES | OrderNames Multi-value text list of fields to order results by – used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition connector-specific syntax conditional clause – used for Select operations | Text |
| LCTOKEN_PROCEDURE | Procedure The stored procedure to execute for the Call method. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------|---|-------------|
| LCTOKEN_RECORD_LIMIT | Result sets produced should be limited to this many records. A value of zero (the default) indicates no limit. | Integer |
| 1 | CommitFrequency Number of data modification actions between commits. A value of zero caused a commit at disconnect; a value of one auto-commits after every action; any other value commits after that many data modification actions. | Integer |

File System Connector Properties

The table below defines the properties for the Lotus Connector for File System.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|---|-------------|
| LCTOKEN_DATABASE | Database Path to the directories (C:\files\somefiles or /unix/usr/files). Required. | Text |
| LCTOKEN_METADATA | Metadata The subdirectory name (mydirectory or somedir/mydir). This subdirectory contains the files you want to access. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNamesMulti-value text list of fields to select - used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Multi-value text list of fields to order results by - used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition The file specification for the files to include. The specification can include the standard wildcard characters: * to match any character string; ? to match any single character. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|--|-------------|
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field – used for Select operations | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value – set by Select operations | Datetime |
| 1 | Binary Whether file contents should be treated as binary (BLOB) data rather than the default (native TEXT). | Boolean |
| 2 | Sort Defines the sort order in the case of ordering by filename (0 = binary, 1 = case sensitive, 2 = case insensitive). | Integer |

File System Metadata

File system metadata is defined as follows:

Filename – Text

Contents – Text

Timestamp – Datetime

Size – Int

ODBC Connector Properties

The table below defines the properties for the Lotus Connector for ODBC.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------|--|-------------|
| LCTOKEN_SERVER | Server Name of the ODBC data source as listed in the ODBC Administrator | Text |
| LCTOKEN_USERID | Userid User ID/name: used for logging in during connection | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|--|-------------|
| LCTOKEN_PASSWORD | Password Password: used for logging in during connection | Text |
| LCTOKEN_METADATA | Metadata Data table source or target for data transfer operations | Text |
| LCTOKEN_INDEX | Index Data index name. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Text list of fields to select – used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Text list of fields to order results by – used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition Connector-specific syntax conditional clause – used for Select operations | Text |
| LCTOKEN_STAMPFIELD | StampField Name of timestamp field used for Select operations | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value - set by Select operations | Datetime |
| LCTOKEN_OWNER | Owner Only objects owned by the owner name in this property are included in the result set | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|-------------------------|---|-------------|
| LCTOKEN_ALT_METADATA | Affects the behavior of METADATA and FIELD catalog types, returning a list of objects of the alternate metadata type instead of the normal metadata type. | Boolean |
| LCTOKEN_CONNECT_TIMEOUT | Maximum number of seconds to wait for a successful connection (the default value of zero indicates to use that connector's default settings). | Integer |
| LCTOKEN_RECORD_LIMIT | Result sets produced should be limited to this many records. A value of zero (the default) indicates no limit. | Integer |
| 1 | CommitFrequency Number of data modification actions between commits. A value of zero causes a commit at disconnect; a value of one auto-commits after every action; any other value commits after that many data modification actions. | Integer |
| 2 | RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state. | Boolean |
| 3 | DisableCursor Whether to force disabling of ODBC cursors. This option reduced performance, and should be used only during writeback operations when the ODBC driver inaccurately reports cursor support, resulting in an error. | Boolean |
| 4 | QuoteColumn String used to surround column names (for example “) | Text |
| 5 | DBMSName Read-only name of the DBMS target of the ODBC connection | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|--|-------------|
| 6 | TraceSQL Select this option to include all SQL statements generated during the execution of the Activity in the Activity's log. This can be helpful when you need to troubleshoot the Activity. | Boolean |
| 7 | SingleThread By default, the Lotus Connector for ODBC is multi-threaded. However, some ODBC drivers are not multi-threaded. Use this property to force the connector into a single thread. | Boolean |

Oracle 7 Connector Properties

The table below defines the properties for the Lotus Connector for Oracle 7.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|-------------------|--|-------------|
| LCTOKEN_SERVER | Server Oracle Host Name. Required. | Text |
| LCTOKEN_USERID | Userid User ID/name – used for logging in during connection | Text |
| LCTOKEN_PASSWORD | Password Password – used for logging in during connection | Text |
| LCTOKEN_METADATA | Metadata Data table or view: the source or target for data transfer operations. | Text |
| LCTOKEN_INDEX | Index Index name. | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------|---|-------------|
| LCTOKEN_FIELDNAMES | FieldNames Text list of fields to select – used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Text list of fields to order results by – used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition Connector-specific syntax conditional clause – used for Select operations | Text |
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field – used for Select operations | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value – set by Select operations | Datetime |
| LCTOKEN_PROCEDURE | Procedure Use when executing parameterized stored procedures in a link-independent fashion. When a session is run and this property is set, the property's value is taken as a stored procedure name and is executed instead of the normal selection operation, potentially producing a result set. If a fieldlist is provided to the Select operation, then all fields in that fieldlist (not just key fields) are supplied to the stored procedure as parameters, with the names taken from the field names. | Text |
| LCTOKEN_OWNER | Owner Only objects owned by the owner name in this property are included in the result set | Text |
| LCTOKEN_ALT_METADATA | Affects the behavior of METADATA and FIELD catalog types, returning a list of objects of the alternate metadata type instead of the normal metadata type. | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------|---|-------------|
| LCTOKEN_CALL_CONTEXT | Indicate what the expected behavior of the procedure call. The default behavior is to execute a procedure and expect (but not require) a result. The following context values may provide more specific information: LCCALL_SELECT: check for (but do not require) a result set. LCCALL_INSERT: Do not check for or produce a result set. LCCALL_UPDATE: Do not check for or produce a result set. LCCALL_REMOVE: Do not check for or produce a result set. LCCALL_COMMAND: Do not check for or produce a result set. LCCALL_NONE: No context information - behavior is defined by the connector. | Integer |
| 1 | CommitFrequency Number of data modification actions between commits. A value of zero causes a commit at disconnect; a value of one auto-commits after every action; any other value commits after that many data modification actions. Note: A commit during a cursored (writeback) operation following an update unlocks result set. Therefore, when a writeback result set is active, the commit frequency property is ignored and commits do not occur. | Integer |
| 2 | RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state. | Boolean |
| 3 | CreateLongColumn Use for table creation. Name of the column to create as a long type during table creation (Oracle tables are restricted to one long column). This value is only used if CreateLongByUser property indicates a user-defined long column. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|--|-------------|
| 4 | CreateLongByUser Use for table creation. Method of selecting the column to create as a long type during table creation. Oracle tables are restricted to no more than one long column. A value of zero indicates that the connector should choose the best candidate from the columns being created (the first and/or longest column). A value of one indicates to use the column name in the CreateLongColumn property. A value of two indicates that no long column should be created. | Integer |
| 5 | TraceSQL Select this option to include all SQL statements generated during the execution of the Activity in the activity's log. This can be helpful when you need to troubleshoot the Activity. | Boolean |

Oracle 8 Connector Properties

The table below defines the properties for the Oracle 8 Connector.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|------------------|---|-------------|
| LCTOKEN_SERVER | Server Oracle Host Name. Required. | Text |
| LCTOKEN_USERID | Userid User ID/name – used for logging in during connection | Text |
| LCTOKEN_PASSWORD | Password Password – used for logging in during connection | Text |
| LCTOKEN_METADATA | Metadata Data table or view: the source or target for data transfer operations. | Text |
| LCTOKEN_INDEX | Index Index name. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|--|-------------|
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Text list of fields to select – used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Text list of fields to order results by – used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition Connector-specific syntax conditional clause – used for Select operations | Text |
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field – used for Select operations | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value – set by Select operations | Datetime |
| LCTOKEN_PROCEDURE | Procedure Use when executing parameterized stored procedures in a link-independent fashion. When a session is run and this property is set, the property's value is taken as a stored procedure name and is executed instead of the normal selection operation, potentially producing a result set. If a fieldlist is provided to the Select operation, then all fields in that fieldlist (not just key fields) are supplied to the stored procedure as parameters, with the names taken from the field names. | Text |
| LCTOKEN_OWNER | Owner Only objects owned by the owner name in this property are included in the result set | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|----------------------|--|-------------|
| LCTOKEN_ALT_METADATA | Affects the behavior of METADATA and FIELD catalog types, returning a list of objects of the alternate metadata type instead of the normal metadata type. | Boolean |
| LCTOKEN_CALL_CONTEXT | Indicate what the expected behavior of the procedure call. The default behavior is to execute a procedure and expect (but not require) a result. The following context values may provide more specific information: LCCALL_SELECT: check for (but do not require) a result set. LCCALL_INSERT: Do not check for or produce a result set. LCCALL_UPDATE: Do not check for or produce a result set. LCCALL_REMOVE: Do not check for or produce a result set. LCCALL_COMMAND: Do not check for or produce a result set. LCCALL_NONE: No context information - behavior is defined by the connector. | Integer |
| 1 | CommitFrequency Number of data modification actions between commits. A value of zero causes a commit at disconnect; a value of one auto- commits after every action; any other value commits after that many data modification actions. Note: A commit during a cursored (writeback) operation following an update unlocks result set. Therefore, when a writeback result set is active, the commit frequency property is ignored and commits do not occur. | Integer |
| 2 | RollbackOnError Whether to rollback the current transaction at disconnection if the session is in an error state. | Boolean |

Note Tokens 3 and 4 are not valid for the Oracle 8 connector.

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|--|-------------|
| 5 | TraceSQL Use this option to include all SQL statements generated during the execution of the Activity in the Activity's log. This can be helpful when you need to troubleshoot the Activity. | Boolean |
| 6 | EnhanceErrorInformation Use this option to include additional error information or warnings in the activities log. | Boolean |
| 7 | DefaultTranslationFormat Fall back text format. It is the name of the Oracle text format to use if automatic translation fails. | Text |

OLE DB Connector Properties

The following table defines the properties for the OLE DB connector.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|---|-------------|
| LCTOKEN_DATABASE | Database – Database name. | Text |
| LCTOKEN_USERID | User ID/name -- User ID/ Name used for logging in during connection. | Text |
| LCTOKEN_PASSWORD | Password – Password. Used for logging in during connection. | Text |
| LCTOKEN_METADATA | Object Class – Data table or view: the source or target for data transfer operations. | Text |
| LCTOKEN_INDEX | Index – Index name. | Text |
| LCTOKEN_MAP_NAME | MapByName – Whether to map by name (TRUE) or position (FALSE) when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback – Whether to perform writeback or non-writeback operations. Even with Writeback operations, the fieldlist must contain key fields which uniquely identify the record to be modified. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames – Text list of fields to select – used for Select operations. | TextList |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|---|-------------|
| LCTOKEN_CONDITION | Condition – OLE DB Provider-specific syntax conditional clause – used for Select operations. | Text |
| LCTOKEN_STAMPFIELD | StampField – Field name of timestamp field – used for Select operations. | Text |
| LCTOKEN_BASESTAMP | BaseStamp – Minimum timestamp value to include in results – used for Select operations. | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp – During a Select operations, set by the connector to the current timestamp value. | Datetime |
| LCTOKEN_PROCEDURE | Procedure – The stored procedure to execute for the Call method. | Datetime |
| LCTOKEN_OWNER | Owner – Only objects owned by the owner name in this property are included in the result set. | Text |
| 1 | Provider – The OLE DB provider's COM programmatic ID (ProgID). Thus it indicates which OLE DB driver to use. For Microsoft SQL Server 7 use "SQLOLEDB"; for Microsoft Access 2000 databases use "Microsoft.Jet.OLEDB.4.0" – Required | Text |
| 2 | Init_Location – Corresponds to the OLE DB DBPROP_INIT_LOCATION property. Not used for either the SQL Server or Jet OLE DB providers. | Text |
| 3 | DataSource – Corresponds to the OLE DB DBPROP_INIT_DATASOURCE property. For the Microsoft SQL Server provider, enter the server name. For the Microsoft Jet provider, use the path to the Jet (.mdb) database. | Text |
| 4 | Init_Catalog – Corresponds to the OLE DB DBPROP_INIT_CATALOG property. For the Microsoft SQL Server provider enter the name of the initial database to use. Not used for the Microsoft Jet provider. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|--|-------------|
| 5 | <p>Init_Mode – Corresponds to the OLE DB DBPROP_INIT_MODE property. Not used for the Microsoft SQL Server provider. For the Microsoft Jet provider, bitwise or the numeric values to indicate special access privileges. The default setting is DB_MODE_READWRITE.</p> <p>The following list shows the Access Privileges, Value, and Description for each setting.</p> <ul style="list-style-type: none"> • DB_MODE_READ – 1 – Indicates read-only • DB_MODE_READWRITE – 3 – Indicates read/write • DB_MODE_SHARE_DENY_WRITE – 8 – Prevents others from opening in write mode • DB_MODE_SHARE_EXCLUSIVE – 12 – Prevents others from opening in read/write mode • DB_MODE_SHARE_DENY_NONE – 16 – Indicates that neither read nor write access can be denied to others | Integer |
| 6 | <p>Init_ProviderString – Corresponds to the OLE DB DBPROP_INIT_PROVIDERSTRING property. OLE DB recognizes an ODBC-like syntax in provider string property values. Within the string, elements are delimited by using a semicolon. The final element in the string must be terminated with a semicolon. Each element consists of a keyword, an equal sign character, and the value passed on initialization, as shown in the following example:</p> <p>Server=Gumby;UID=everett;</p> <p>For the Microsoft SQL Server provider, developers may need to set the network property to the name of the Net-Library (DLL) used to communicate with the SQL Server. The name should not include the path or the .dll file name extension. So for example to use the TCP/IP network library, add the following: "Network=DBMSSOCN;"</p> <p>Alternatively, the default Net-Library can be set by the SQL Server Client Network Utility.</p> | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 7 | <p>Auth_Integrated – Corresponds to the OLE DB DBPROP_AUTH_INTEGRATED property. Indicates the authentication service to use if any. For the Microsoft SQL Server provide, one can enter "SSPI" in order to use Windows NT Authentication service for authorizing user access to the SQL Server database. If this property is not set, the SQL Server login and password should be specified in the UserID and Password properties.</p> <p>Not used for the Microsoft Jet provider.</p> | Text |
| 8 | <p>CommitFrequency – Number of data modification actions between commits. A value of zero causes a commit at disconnect; a value of one auto-commits every action; any other value commits after that many data modification actions.</p> | Integer |
| 9 | <p>IsolationLevel – This property determines the extent that outside actions can affect a transaction. Read Committed is the default option.</p> <p>The following list shows the Name, Description, and Value for each option.</p> <ul style="list-style-type: none"> • Read Uncommitted – A transaction operating at the Read Uncommitted level can see uncommitted changes made by other transactions. At this level of isolation, dirty reads, nonrepeatable reads, and phantoms are all possible – 256 • Read Committed – A transaction operating at the Read Committed level cannot see changes made by other transactions until those transactions are committed. At this level of isolation, dirty reads are not possible but nonrepeatable reads and phantoms are possible – 4,096 | Integer |

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| | <ul style="list-style-type: none"> • Repeatable Read – A transaction operating at the Repeatable Read level is guaranteed not to see any changes made by other transactions in values it has already read. At this level of isolation, dirty reads and nonrepeatable reads are not possible but phantoms are possible. – 65,536 • Serializable – A transaction operating at the Serializable level guarantees that all concurrent transactions interact only in ways that produce the same effect as if each transaction were entirely executed one after the other. At this isolation level, dirty reads, nonrepeatable reads, and phantoms are not possible. transactions interact only in ways that produce the same effect as if each transaction were entirely executed one after the other. At this isolation level, dirty reads, nonrepeatable reads, and phantoms are not possible. – 1,048,576 | |
| 10 | TraceSQL – Include all SQL statements generated during the execution of the activity in the activity's log. | Boolean |

Sybase Connector Properties

The table below defines the properties for the Lotus Connector for Sybase.

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|------------------|--|-------------|
| LCTOKEN_SERVER | Server Server name. Required. | Text |
| LCTOKEN_DATABASE | Database Database name. | Text |
| LCTOKEN_USERID | User ID/name Used for logging in during connection | Text |
| LCTOKEN_PASSWORD | Password Password – used for logging in during connection | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------------|---|-------------|
| LCTOKEN_METADATA | Metadata Metadata object – the source or target for data transfer operations. A form (Notes), table (relational DB), or other data container | Text |
| LCTOKEN_INDEX | Index Index name. A view (Notes), index (relational DB), or other data index | Text |
| LCTOKEN_MAP_NAME | MapByName Whether to map by name or position when transferring data between data source and target. | Boolean |
| LCTOKEN_WRITEBACK | Writeback Whether to perform writeback or non-writeback operations. | Boolean |
| LCTOKEN_FIELDNAMES | FieldNames Multi-value text list of fields to select – used for Select operations | Text List |
| LCTOKEN_ORDERNAMES | OrderNames Multi-value text list of fields to order results by – it is used for Select operations | Text List |
| LCTOKEN_CONDITION | Condition Connector-specific syntax conditional clause – used for Select operations | Text |
| LCTOKEN_STAMPFIELD | StampField Field name of timestamp field – used for Select operations | Text |
| LCTOKEN_BASESTAMP | BaseStamp Minimum timestamp value to include in results – used for Select operations | Datetime |
| LCTOKEN_MAXSTAMP | MaxStamp During a Select operations, set by the connector to the current timestamp value - set by Select operations | Datetime |
| LCTOKEN_PROCEDURE | Procedure The stored procedure to execute for the Call method. Any returned result set is available. | Text |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|-------------------------|---|-------------|
| LCTOKEN_OWNER | Owner Only objects owned by the owner name in this property are included in the result set. | Text |
| LCTOKEN_ALT_METADATA | Affects the behavior of METADATA and FIELD catalog types, returning a list of objects of the alternate metadata type instead of the normal metadata type. | Boolean |
| LCTOKEN_CONNECT_TIMEOUT | Maximum number of seconds to wait for a successful connection (the default value of zero indicates to use that connector's default settings). | Integer |
| LCTOKEN_RECORD_LIMIT | Result sets produced should be limited to this many records. A value of zero (the default) indicates no limit. | Integer |
| 3 | CreateShortBound Whether to truncate bounded text and binary data during table creation. When used, all text and binary fields with a maximum length will be truncated to fit in a char/varchar/binary/ varbinary column. | Boolean |
| 4 | CreateShortUnbound Whether to truncate unbounded text and binary data during table creation. When used, all text and binary fields with no maximum length will be truncated to fit in a char/varchar/binary/ varbinary column. | Boolean |
| 5 | DisableCursor Disable Sybase writeback cursors. | Boolean |

continued

| <i>Token</i> | <i>Name/Description</i> | <i>Type</i> |
|--------------|---|-------------|
| 6 | <p>ProcedureStatus</p> <p>A Sybase stored procedure can either generate a result set or return a status code. When a non-zero code is returned, this may signal an error. The default behavior is to interpret the status as a 1-row, 1-column result set. Selecting this option interprets the status code separately, not as a result set (non-zero is error, and zero is success).</p> | Boolean |
| 7 | <p>TraceSQL</p> <p>Select this option to include all SQL statements generated during the execution of the Activity in the Activity's log. This can be helpful when you need to troubleshoot the activity.</p> | Boolean |

Appendix D

Character Sets

This appendix provides information about character sets, including character set translation, character sort order, and a list of the character sets supported by the LotusScript Extensions Language (LC LSX and LEI LSX).

List of Supported Character Sets

The list is given as text stream format constants. Any of these values may be used as a stream format for a text stream when creating scripts using the Lotus Connectors LotusScript Extensions. To indicate the character set on the local machine, use the constant LCSTREAMFMT_NATIVE.

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|---|
| LCSTREAMFMT_LICS | Lotus International Character Set |
| LCSTREAMFMT_IBMCP851 | MS-DOS PC Greek (CP 851) |
| LCSTREAMFMT_IBMCP852 | MS-DOS PC Eastern European (CP 852) |
| LCSTREAMFMT_IBMCP853 | MS-DOS PC Turkish (CP 853) |
| LCSTREAMFMT_IBMCP857 | MS-DOS PC Turkish (CP 857) |
| LCSTREAMFMT_IBMCP862 | MS-DOS PC Hebrew (CP 862) |
| LCSTREAMFMT_IBMCP864 | MS-DOS PC Arabic (CP 864) |
| LCSTREAMFMT_IBMCP866 | MS-DOS PC Cyrillic Unicode (CP 866) |
| LCSTREAMFMT_IBMCP437 | MS-DOS PC US (CP 437) |
| LCSTREAMFMT_IBMCP850 | MS-DOS PC Western European (CP 850) |
| LCSTREAMFMT_IBMCP855 | MS-DOS PC Cyrillic (CP 855) |
| LCSTREAMFMT_IBMCP860 | MS-DOS PC Portuguese (CP 860) |
| LCSTREAMFMT_IBMCP861 | MS-DOS PC Icelandic (CP 861) |
| LCSTREAMFMT_IBMCP863 | MS-DOS PC Canadian French (CP 863) |
| LCSTREAMFMT_IBMCP865 | MS-DOS PC Norwegian (CP 865) |
| LCSTREAMFMT_IBMCP869 | MS-DOS PC Greek (CP 869) |
| LCSTREAMFMT_IBMCP899 | IBM Code Page 899 (CP 899) |
| LCSTREAMFMT_IBMCP932 | MS-DOS PC Japanese Microsoft Shift-JIS (CP 932) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|---|
| LCSTREAMFMT_IBMCP942 | MS-DOS PC Japanese Microsoft Shift-JIS (CP 942) |
| LCSTREAMFMT_IBMCP891 | MS-DOS PC Korean (CP 891) |
| LCSTREAMFMT_DECMCS | DEC Multinational Character Set |
| LCSTREAMFMT_EUC | Extended Unix Code |
| LCSTREAMFMT_KS | MS-DOS Korean – KSC 5601 |
| LCSTREAMFMT_IBMCP949 | MS-DOS Korean (CP 949) |
| LCSTREAMFMT_TCA | TCA |
| LCSTREAMFMT_BIG5 | MS-DOS Taiwan (traditional) Chinese (BIG-5) |
| LCSTREAMFMT_IBMCP950 | MS-DOS Taiwan (traditional) Chinese (CP 950) |
| LCSTREAMFMT_GB | MS-DOS PRC (simplified) Chinese (GB 2312) |
| LCSTREAMFMT_IBMCP936 | MS-DOS PRC (simplified) Chinese (CP 936) |
| LCSTREAMFMT_NECESJIS | MS-DOS PC Japanese NEC Shift-JIS (CP 932) |
| LCSTREAMFMT_ISO646 | ASCII |
| LCSTREAMFMT_ASCII | ASCII |
| LCSTREAMFMT_ISO88591 | ISO Latin-1 US, Western European (ISO-8859-1) |
| LCSTREAMFMT_IBMCP819 | ISO Latin-1 US, Western European (CP 819) |
| LCSTREAMFMT_ISO88592 | ISO Latin-2 Eastern European (ISO-8859-2) |
| LCSTREAMFMT_IBMCP912 | ISO Latin-2 Eastern European (CP 912) |
| LCSTREAMFMT_ISO88593 | ISO Latin-3 Southern European (ISO-8859-3) |
| LCSTREAMFMT_ISO88594 | ISO Latin-4 Northern European (ISO-8859-4) |
| LCSTREAMFMT_ISO88595 | ISO Cyrillic (ISO-8859-5) |
| LCSTREAMFMT_IBMCP915 | ISO Cyrillic (CP 915) |
| LCSTREAMFMT_ISO88596 | ISO Arabic (ISO-8859-6) |
| LCSTREAMFMT_IBMCP1008 | ISO Arabic (CP 1008) |
| LCSTREAMFMT_ISO88597 | ISO Greek (ISO-8859-7) |
| LCSTREAMFMT_IBMCP813 | ISO Greek (CP 813) |
| LCSTREAMFMT_ISO88598 | ISO Hebrew (ISO-8859-8) |
| LCSTREAMFMT_IBMCP916 | ISO Hebrew (CP 916) |
| LCSTREAMFMT_ISO88599 | ISO Latin-5 Southern European (ISO-8859-9) |
| LCSTREAMFMT_IBMCP920 | ISO Latin-5 Southern European (CP 920) |
| LCSTREAMFMT_HPROMAN | HP Roman (LaserJet) |
| LCSTREAMFMT_HPGREEK | HP Greek (LaserJet) |
| LCSTREAMFMT_HPTURKISH | HP Turkish (LaserJet) |
| LCSTREAMFMT_HPHEBREW | HP Hebrew (LaserJet) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|---|
| LCSTREAMFMT_HPARABIC | HP Arabic (LaserJet) |
| LCSTREAMFMT_HPTHAI | HP Thai (LaserJet) |
| LCSTREAMFMT_HPJAPAN | HP Japanese (LaserJet) |
| LCSTREAMFMT_HPKANA | HP Kana (LaserJet) |
| LCSTREAMFMT_HPKOREA | HP Korean (LaserJet) |
| LCSTREAMFMT_HPPRC | HP Simplified Chinese (LaserJet) |
| LCSTREAMFMT_HPROC | HP Traditional Chinese (LaserJet) |
| LCSTREAMFMT_IBMCP37 | IBM EBCDIC US/Canadian English (CP 37) |
| LCSTREAMFMT_IBMCP28709 | IBM Code Page 28709 (CP28709) |
| LCSTREAMFMT_IBMCP273 | IBM EBCDIC German – Austrian (CP 273) |
| LCSTREAMFMT_IBMCP278 | IBM EBCDIC Finnish, Swedish (CP 278) |
| LCSTREAMFMT_IBMCP280 | IBM EBCDIC Italian (CP 280) |
| LCSTREAMFMT_IBMCP284 | IBM EBCDIC Spanish, Latin American (CP 284) |
| LCSTREAMFMT_IBMCP285 | IBM EBCDIC UK (CP 285) |
| LCSTREAMFMT_IBMCP290 | IBM EBCDIC Japanese (Katakana) (CP 290) |
| LCSTREAMFMT_IBMCP297 | IBM EBCDIC French (CP 297) |
| LCSTREAMFMT_IBMCP500 | IBM EBCDIC International (CP 500) |
| LCSTREAMFMT_IBMCP277 | IBM EBCDIC Danish, Norwegian (CP 277) |
| LCSTREAMFMT_IBMCP1047 | IBM EBCDIC Latin-1 Open Systems (CP 1047) |
| LCSTREAMFMT_IBMCP1250 | Windows Eastern European (CP 1250) |
| LCSTREAMFMT_IBMCP1251 | Windows Cyrillic (CP 1251) |
| LCSTREAMFMT_IBMCP1252 | Windows ANSI (CP 1252) |
| LCSTREAMFMT_ANSI | ANSI |
| LCSTREAMFMT_IBMCP1253 | Windows Greek (CP 1253) |
| LCSTREAMFMT_IBMCP1254 | Windows Turkish (CP 1254) |
| LCSTREAMFMT_IBMCP1255 | Windows Hebrew (CP 1255) |
| LCSTREAMFMT_IBMCP1256 | Windows Arabic (CP 1256) |
| LCSTREAMFMT_IBMCP1257 | Windows Baltic (CP 1257) |
| LCSTREAMFMT_IBMCP1363 | Windows Korean (CP 1363) |
| LCSTREAMFMT_MACSCRIPT0 | Macintosh Roman (Script 0) |
| LCSTREAMFMT_MACSCRIPT1 | Macintosh Japanese (Script 1) |
| LCSTREAMFMT_MACSCRIPT2 | Macintosh Traditional Chinese (Script 2) |
| LCSTREAMFMT_MACSCRIPT3 | Macintosh Korean (Script 3) |
| LCSTREAMFMT_MACSCRIPT4 | Macintosh Arabic (Script 4) |
| LCSTREAMFMT_MACSCRIPT5 | Macintosh Hebrew (Script 5) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|---------------------------------|--|
| LCSTREAMFMT_MACSCRIPT6 | Macintosh Greek (Script 6) |
| LCSTREAMFMT_MACSCRIPT7 | Macintosh Cyrillic (Script 7) |
| LCSTREAMFMT_MACSCRIPT8 | Macintosh Right-left symbol (Script 8) |
| LCSTREAMFMT_MACSCRIPT9 | Macintosh Devanagari (Script 9) |
| LCSTREAMFMT_MACSCRIPT10 | Macintosh Gurmukhi (Script 10) |
| LCSTREAMFMT_MACSCRIPT11 | Macintosh Gujarati (Script 11) |
| LCSTREAMFMT_MACSCRIPT12 | Macintosh Oriya (Script 12) |
| LCSTREAMFMT_MACSCRIPT13 | Macintosh Bengali (Script 13) |
| LCSTREAMFMT_MACSCRIPT14 | Macintosh Tamil (Script 14) |
| LCSTREAMFMT_MACSCRIPT15 | Macintosh Telugu (Script 15) |
| LCSTREAMFMT_MACSCRIPT16 | Macintosh Kannada/Kanarese (Script 16) |
| LCSTREAMFMT_MACSCRIPT17 | Macintosh Malayalam (Script 17) |
| LCSTREAMFMT_MACSCRIPT18 | Macintosh Sinhalese (Script 18) |
| LCSTREAMFMT_MACSCRIPT19 | Macintosh Burmese (Script 19) |
| LCSTREAMFMT_MACSCRIPT20 | Macintosh Khmer/Cambodian (Script 20) |
| LCSTREAMFMT_MACSCRIPT21 | Macintosh Thai (Script 21) |
| LCSTREAMFMT_MACSCRIPT22 | Macintosh Laotian (Script 22) |
| LCSTREAMFMT_MACSCRIPT23 | Macintosh Georgian (Script 23) |
| LCSTREAMFMT_MACSCRIPT24 | Macintosh Armenian (Script 24) |
| LCSTREAMFMT_MACSCRIPT25 | Macintosh Simplified Chinese (Script 25) |
| LCSTREAMFMT_MACSCRIPT26 | Macintosh Tibetan (Script 26) |
| LCSTREAMFMT_MACSCRIPT27 | Macintosh Mongolian (Script 27) |
| LCSTREAMFMT_MACSCRIPT28 | Macintosh Geez/Ethiopic (Script 28) |
| LCSTREAMFMT_MACSCRIPT29 | Macintosh EastEurRoman/Slavic (Script 29) |
| LCSTREAMFMT_MACSCRIPT30 | Macintosh Vietnamese (Script 30) |
| LCSTREAMFMT_MACSCRIPT31 | Macintosh extended Arabic/Sindhi (Script 31) |
| LCSTREAMFMT_MACSCRIPT32 | Macintosh un-interpreted symbols (Script 32) |
| LCSTREAMFMT_MACSCRIPT0CROATIAN | Macintosh Roman variant – Croatian |
| LCSTREAMFMT_MACSCRIPT0GREEK | Macintosh Roman variant – Greek |
| LCSTREAMFMT_MACSCRIPT0ICELANDIC | Macintosh Roman variant – Icelandic |
| LCSTREAMFMT_MACSCRIPT0ROMANIAN | Macintosh Roman variant – Romanian |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|----------------------------------|--|
| LCSTREAMFMT_MACSCRIPT0TURKISH | Macintosh Roman variant – Turkish |
| LCSTREAMFMT_THAI | MS Thai Windows |
| LCSTREAMFMT_IBMCP874 | MS-DOS PC Thai (CP 874) |
| LCSTREAMFMT_ISO885911 | ISO Thai (ISO-8859-11) |
| LCSTREAMFMT_TIS620 | Thai Industrial Standard (TIS620-2529) |
| LCSTREAMFMT_UNICODE | Unicode (ISO 10646) |
| LCSTREAMFMT_IBMCP1200 | Unicode (IBM CP 1200) |
| LCSTREAMFMT_ISO10646 | Unicode (ISO 10646) |
| LCSTREAMFMT_UTF7 | Unicode Transformation Formats 7 |
| LCSTREAMFMT_UTF8 | Unicode Transformation Formats 8 |
| LCSTREAMFMT_LMBCS | Lotus MultiByte Character Set (LMBCS) |
| LCSTREAMFMT_DECNRCUK | DEC National Replacement Char – UK |
| LCSTREAMFMT_DECNRCDUTCH | DEC Nat'l Replacement Char – Dutch |
| LCSTREAMFMT_DECNRCFINNISH | DEC Nat'l Replacement Char – Finnish |
| LCSTREAMFMT_DECNRCFRENCH | DEC Nat'l Replacement Char – French |
| LCSTREAMFMT_DECNRCFRENCHCANADIAN | DEC Nat'l Replacement Char – French Canadian |
| LCSTREAMFMT_DECNRCGERMAN | DEC Nat'l Replacement Char – German |
| LCSTREAMFMT_DECNRCITALIAN | DEC Nat'l Replacement Char – Italian |
| LCSTREAMFMT_DECNRCNORWEGIAN | DEC Nat'l Replacement Char – Norwegian |
| LCSTREAMFMT_DECNRCNORWEGIAN | Danish |
| LCSTREAMFMT_DECNRCPORTUGUESE | DEC Nat'l Replacement Char – Portuguese |
| LCSTREAMFMT_DECNRCSPANISH | DEC Nat'l Replacement Char – Spanish |
| LCSTREAMFMT_DENRCSWEDISH | DEC Nat'l Replacement Char – Swedish |
| LCSTREAMFMT_DENRCSWISS | DEC Nat'l Replacement Char – Swiss |
| LCSTREAMFMT_T61 | Teletex T.61 |
| LCSTREAMFMT_T50 | Teletex T.50 |
| LCSTREAMFMT_ASN1 | ANSI Standard Notation (ASN.1) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|--|
| LCSTREAMFMT_IBMCP856 | MS-DOS PC Hebrew (CP 85) |
| LCSTREAMFMT_IBMCP1004 | MS-DOS PC Desktop Publishing (CP 1004) |
| LCSTREAMFMT_IBMCP1002 | IBM EBCDIC DCF (CP 1002) |
| LCSTREAMFMT_IBMCP1003 | IBM EBCDIC US Text Subset (CP 1003) |
| LCSTREAMFMT_IBMCP1025 | IBM EBCDIC Cyrillic (CP 1025) |
| LCSTREAMFMT_IBMCP1026 | IBM EBCDIC Turkish (CP 1026) |
| LCSTREAMFMT_IBMCP1028 | IBM EBCDIC Hebrew Publishing (CP 1028) |
| LCSTREAMFMT_IBMCP256 | IBM EBCDIC International #1 (CP 256) |
| LCSTREAMFMT_IBMCP259 | IBM EBCDIC Symbols Set 7 (CP 259) |
| LCSTREAMFMT_IBMCP274 | IBM EBCDIC Belgian (CP 274) |
| LCSTREAMFMT_IBMCP275 | IBM EBCDIC Brazilian (CP 275) |
| LCSTREAMFMT_IBMCP281 | IBM EBCDIC Japanese (Latin) (CP 281) |
| LCSTREAMFMT_IBMCP282 | IBM EBCDIC Portuguese (CP 282) |
| LCSTREAMFMT_IBMCP361 | IBM EBCDIC International #5 (CP 361) |
| LCSTREAMFMT_IBMCP382 | IBM EBCDIC Austrian, German, Switzerland (CP 382) |
| LCSTREAMFMT_IBMCP383 | IBM EBCDIC Belgian (CP 383) |
| LCSTREAMFMT_IBMCP384 | IBM EBCDIC Brazilian (CP 384) |
| LCSTREAMFMT_IBMCP385 | IBM EBCDIC Canadian (French) (CP 385) |
| LCSTREAMFMT_IBMCP386 | IBM EBCDIC Danish, Norwegian (CP 386) |
| LCSTREAMFMT_IBMCP387 | IBM EBCDIC Finnish, Swedish (CP 387) |
| LCSTREAMFMT_IBMCP388 | IBM EBCDIC French, Swiss (CP 388) |
| LCSTREAMFMT_IBMCP389 | IBM EBCDIC Italian, Swiss (CP 389) |
| LCSTREAMFMT_IBMCP390 | IBM EBCDIC Japanese (Latin) (CP 390) |
| LCSTREAMFMT_IBMCP391 | IBM EBCDIC Portuguese (CP 391) |
| LCSTREAMFMT_IBMCP392 | IBM EBCDIC Spanish, Philippines (CP 392) |
| LCSTREAMFMT_IBMCP393 | IBM EBCDIC Latin American (Spanish Speaking) (CP 393) |
| LCSTREAMFMT_IBMCP394 | IBM EBCDIC UK, Australian, Hong Kong, Ireland, New Zealand (CP 394) |
| LCSTREAMFMT_IBMCP395 | IBM EBCDIC US, Canadian (English) (CP 395) |
| LCSTREAMFMT_IBMCP423 | IBM EBCDIC Greek 183 (CP 423) |
| LCSTREAMFMT_IBMCP424 | IBM EBCDIC Hebrew (CP 424) |
| LCSTREAMFMT_IBMCP803 | IBM EBCDIC Hebrew Character Set A (CP 803) |
| LCSTREAMFMT_IBMCP870 | IBM EBCDIC Eastern Europe (CP 870) |
| LCSTREAMFMT_IBMCP871 | IBM EBCDIC Icelandic (CP 871) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|--|
| LCSTREAMFMT_IBMCP875 | IBM EBCDIC Greek (CP 875) |
| LCSTREAMFMT_IBMCP880 | IBM EBCDIC Cyrillic (CP 880) |
| LCSTREAMFMT_IBMCP905 | IBM EBCDIC Turkish (CP 905) |
| LCSTREAMFMT_IBMCP948 | IBM Extended Taiwanese (CP 948) |
| LCSTREAMFMT_IBMCP938 | IBM Taiwanese (CP 938) |
| LCSTREAMFMT_IBMCP1381 | IBM GBK = GB + Hanzi (CP 1381) |
| LCSTREAMFMT_IBMCP1386 | IBM Traditional Chinese (CP 1386) |
| LCSTREAMFMT_EACC | East Asian Character Code Set (ANSI Z39.64-1989) |
| LCSTREAMFMT_JIS | Japanese Information Standard 0201 (JIS 201) |
| LCSTREAMFMT_CCCII | Chinese Character Code for Information Interchange (Taiwan) |
| LCSTREAMFMT_XEROXCJK | Xerox CJK |
| LCSTREAMFMT_IBMCP944 | IBM Extended Korean (CP 944) |
| LCSTREAMFMT_IBMCP934 | IBM Korean (CP 934) |
| LCSTREAMFMT_IBMCP737 | MS-DOS PC Greek (CP 737) |
| LCSTREAMFMT_IBMCP775 | MS-DOS PC Baltic (CP 775) |
| LCSTREAMFMT_ISO6937 | Latin chars (non-spacing accents) similar to T.61 |
| LCSTREAMFMT_BASE64 | Content-Transfer-Encoding |
| LCSTREAMFMT_JIS2 | Japanese Information Standard 0208 (JIS 208) |
| LCSTREAMFMT_EUCJ | Extended Unix Code – Japanese |
| LCSTREAMFMT_EUCT | Extended Unix Code – Taiwanese |
| LCSTREAMFMT_EUCK | Extended Unix Code – Korean |
| LCSTREAMFMT_ISOKR | ISO-2022-KR switching; treated as EUCK |
| LCSTREAMFMT_EUCC | Extended Unix Code – Chinese |
| LCSTREAMFMT_IBMCP921 | Replacement for Lithuanian (CP 921) |
| LCSTREAMFMT_IBMCP922 | Russian (CP 922) |
| LCSTREAMFMT_KOI8 | Cyrillic Internet Support |
| LCSTREAMFMT_IBMCP720 | IBM Code Page 720 (CP 720) |
| LCSTREAMFMT_IBMCP1258 | Windows Vietnamese (CP 1258) |
| LCSTREAMFMT_ISO885910 | ISO Latin-6 (ISO-8859-10) |
| LCSTREAMFMT_JP1TEXT | OSI/JIS X 5003-1987 X.400 Japanese ISP |
| LCSTREAMFMT_VIQRI | Vietnamese Quoted Readable |
| LCSTREAMFMT_VISCII | Vietnamese VISCII 1.1 (VICSII) |
| LCSTREAMFMT_VISCII1 | TCVN Vietnamese Orthographic (VCSII-1) |
| LCSTREAMFMT_VISCII2 | TCVN Vietnamese Graphic (VCSII-2)* / |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|--|
| LCSTREAMFMT_IBMCP838 | IBM EBCDIC SBCS Thai (CP 838) |
| LCSTREAMFMT_IBMCP9030 | IBM EBCDIC SBCS Thai (CP 9030) |
| LCSTREAMFMT_IBMCP833 | IBM EBCDIC SBCS Korean – extended (CP 833) |
| LCSTREAMFMT_IBMCP836 | IBM EBCDIC SBCS PRC (simplified) Chinese (CP 836) |
| LCSTREAMFMT_IBMCP1027 | IBM EBCDIC SBCS Japanese Latin – extended (CP 1027) |
| LCSTREAMFMT_IBMCP420 | IBM EBCDIC Arabic (CP 420) |
| LCSTREAMFMT_IBMCP918 | IBM EBCDIC Code Page 918 (CP 918) |
| LCSTREAMFMT_IBMCP1097 | IBM EBCDIC Code Page 1097 (CP 1097) |
| LCSTREAMFMT_IBMCP1112 | IBM EBCDIC Code Page 1112 (CP 1112) |
| LCSTREAMFMT_IBMCP1122 | IBM EBCDIC Code Page 1122 (CP 1122) |
| LCSTREAMFMT_IBMCP1123 | IBM EBCDIC Code Page 1123 (CP 1123) |
| LCSTREAMFMT_IBMCP1129 | IBM EBCDIC Code Page 1129 (CP 1129) |
| LCSTREAMFMT_IBMCP1130 | IBM EBCDIC Code Page 1130 (CP 1130) |
| LCSTREAMFMT_IBMCP1132 | IBM EBCDIC Code Page 1132 (CP 1132) |
| LCSTREAMFMT_IBMCP1133 | IBM EBCDIC Code Page 1133 (CP 1133) |
| LCSTREAMFMT_IBMCP930 | IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 930) |
| LCSTREAMFMT_IBMCP933 | IBM EBCDIC EUC Korean Mixed (CP 933) |
| LCSTREAMFMT_IBMCP935 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 935) |
| LCSTREAMFMT_IBMCP937 | IBM EBCDIC EUC Taiwan (traditional) Chinese Mixed (CP 937) |
| LCSTREAMFMT_IBMCP939 | IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 939) |
| LCSTREAMFMT_IBMCP931 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 931) |
| LCSTREAMFMT_IBMCP1388 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 1388) |
| LCSTREAMFMT_IBMCP5026 | IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 5026) |
| LCSTREAMFMT_IBMCP5035 | IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 5035) |
| LCSTREAMFMT_IBMCP300 | IBM EBCDIC DBCS Japanese (CP 300) |
| LCSTREAMFMT_IBMCP834 | IBM EBCDIC DBCS Korean (CP 834) |
| LCSTREAMFMT_IBMCP835 | IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 835) |

continued

| <i>Stream Format Constant</i> | <i>Description</i> |
|-------------------------------|--|
| LCSTREAMFMT_IBMCP837 | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 837) |
| LCSTREAMFMT_IBMCP930X | IBM EBCDIC DBCS Japanese (CP 930X) |
| LCSTREAMFMT_IBMCP933X | IBM EBCDIC DBCS Korean (CP 933X) |
| LCSTREAMFMT_IBMCP935X | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 935X) |
| LCSTREAMFMT_IBMCP937X | IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 937X) |
| LCSTREAMFMT_IBMCP939X | IBM EBCDIC DBCS Japanese (CP 939X) |
| LCSTREAMFMT_IBMCP931X | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 931X) |
| LCSTREAMFMT_IBMCP1388X | IBM EBCDIC DBCS PRC (CP 1388X) |
| LCSTREAMFMT_IBMCP1383 | IBM Traditional Chinese (CP 1383) |
| LCSTREAMFMT_IBMCP806 | ISO Devnagiri (CP 806) |
| LCSTREAMFMT_IBMCP1137 | IBM EBCDIC Devnagiri (CP 1137) |
| LCSTREAMFMT_VISCII3 | TCVN3 Vietnamese (VCSII-3) |
| LCSTREAMFMT_TCVN3 | TCVN3 Vietnamese (VCSII-3) |

Appendix E

Additional Scripting Capability within LEI LSX

This appendix provides information about additional methods and properties provided with LEI LSX LSession class that expand on the LotusScript Extensions for Domino Connector Classes (LSX) described in this manual. These additional LSession class methods available only with LEI LSX.

The following LSession methods are described in this appendix:

- AddProperty
- GetCommand
- GetProperty
- GetProperty<Type>
- ListProperty
- LookupProperty
- Log
- LogEx
- LogText
- LogTextEx
- RunActivity
- SetProperty
- SetProperty<Type>

Overview of Expanded Classes

The LEI LSX contains all the current functionality of Domino/DECS LC LSX and more. However, as of LEI 3.1, LEI no longer uses the LC LSX; it instead uses the new LEI LSX. If you have LEI 3.1 loaded, it automatically uses the (n)lsxlei.dll for LSX scripting; not the (n)lsxlc.dll. Prior to LEI 3.1, LEI used LC LSX and the (n)lsxlc.dll.

Note If you are an existing user and have LotusScripts that contain the term `lsxlc` (for example, `Uselsx "lsxlc"`), please rename all instances of `lsxlc` with `lsxlei` in your scripts. Editing existing scripts to call the `lsxlei` (for example, `Uselsx "lsxlei"`) will ensure that the correct API is used.

To invoke LEI LSX, enter the following statement:

```
Uselsx "lsxlei"
```

In addition, you must do the following:

- Dim `LCSession` as a named session (this becomes the default Log Document name)
- `LCConnections` must be dimensioned as existing named connection documents that have been created in the LEI Administrator.

See the following example.

Options

```
Uselsx "lsxlei"
```

Sub Initialize

```
Dim MyLEISession As New LCSession ("MainSession")
```

```
Dim MyNotesConnection as New LCConnection ("MyNotesConn")
```

```
Dim MyOracleConnection as New LCConnection  
("MyOracleConn")
```

Note In the above example, “MyNotesConn” and “MyOracleConn” are named connection documents in the LEI Administrator.

Expanded LCSession Functionality

The LEI LSX expands the `LCSession` Class functionality with properties not available with LC LSX. The `LCSession` class also provides additional methods to support activity-level properties and logging. The additional features of the Domino Connector Classes for LEI are only available when used in conjunction with LEI and the LEI Administrator.

The use of the LEI Administrator and logging is determined when the `LCSession` class object is created. LEI Scripted activities require that the first Domino Connector Class object created be an `LCSession` object.

When an LEI Scripted Activity is run, all the property fields of the activity form are accessible in the script. Additionally, when a Scripted Activity is used in conjunction with the LEI CGI program and an HTML form, the form’s field values appear in the session object as Read-Only properties.

`LCSession` properties may be accessed through several techniques.

- After creating the LCSession object, you may specify the property. For example, if session is the name of a LCSession object and CharacterSet is a property of the session, session.CharacterSet is the value of the item. When using the Notes LotusScript development environment, the debugger displays all session properties when inspecting variables.
- You can use the GetProperty and SetProperty methods of LCSession to access the property as an LCField object.
- You can use GetProperty<type> and SetProperty<type> to access the property as a specific data type.

Named Sessions, Named Connection Documents, and the LEI LSX

As of LEI release 3.1, the LEI LSX requires that you provide a name for every session and that you specify connection documents. Example statement syntax is shown below:

```
Useslsx "*lsxlei"
```

```
Dim mySession As New LCSession ("MyLEISession")
```

```
Dim myOracleConnection as New LCConnection ("OracleConnDoc")
```

where MyLEISession is the name that LEI uses to create an entry in the LEI Log database each time the script is executed, and OracleConnDoc is the name of an existing connection document in the LEI Administrator.

Note One benefit of named sessions is that they enable the LEI LSX to support activity log documents.

Caution Prior to LEI release 3.1, LEI used the LC LSX, which did not support a name argument in the Dim x as new LCSession statement. Existing LEI users should make note of this new behavior and provide a name for all LCSessions.

The LEI Administrator uses the LEI LSX to perform various browsing calls, such as Select Matadata and Map Fields.

AddProperty Method for LCSession

This method creates a new property and assigns it a property token.

Defined In
LCSession Class

Syntax
tokenID = lcSession.AddProperty (propertyName, storageType

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|--|
| <i>propertyName</i> | String. Name of the property to be created. The name must be unique among properties of this storage type. Using the same name as an existing property creates a new token if the storage type differs, but simply returns the previously assigned token if the storage type is the same. |
| <i>storageType</i> | Long. Optional. One of the following: LCPROPSTORE_STATIC This value is read from or written to the activity document. LCPROPSTORE_DYNAMIC This value will only exist during this run. LCPROPSTORE_SYSTEM This value is read from and written to operating system environment variables. Note System properties are not persistent across executions. |

Return Value

| <i>Value</i> | <i>Description</i> |
|----------------|---|
| <i>tokenID</i> | Token assigned to the property. The token is only valid for this run of the activity. The token will be a value between the highest token defined by the activity itself and the maximum base token value, 65535. |

Comment

Any added properties are in addition to the properties defined by an activity. There are three types of properties which may be added. Static properties will be stored in the activity document itself in the LEI Administrator database, and their values may be retrieved for future executions of the same activity. Dynamic properties are defined only for the current execution of the activity, and are discarded when the activity terminates. System properties are represented by operating system environment variables, and any retrieval or assignment of values immediately alters the environment variable's value.

When the LEI activity is initiated from the command line, the command line parameters are added into the activity as dynamic properties. The entire command line becomes a text list property with the name "". If the activity is executed as a CGI program, then the CGI parameters are parsed and then also added as dynamic properties.

GetCommand Method for LCSession

This method obtains any command supplied to the activity by LEI.

Defined in

LCSession Class

Syntax

```
GetCommand () As NPSTATUS
```

Parameters

None

Return Value

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>(routine)</i> | Status command value returned. A status of LCSUCCESS indicates that no action needs to be taken. If the status is non-zero, then it is one of the following values: LCFAIL_ACTIVITY_SHUTDOWN: Terminate the activity due to a close command. LCFAIL_ACTIVITY_TIMEOUT: Terminate the activity due to a timeout. |

Comment

Commands are sent when a request for an orderly shutdown of an activity occurs due to user request or activity timeout. No commands are sent for activities which were started outside the control of LEI rather than through the LEI Server. Any command is automatically checked by all Connect functions, but should also be checked periodically during any non-trivial processing in an activity or script.

GetProperty Method for LCSession

This method obtains a copy of the current value for an activity property.

Defined In

LCSession Class

Syntax

```
destField = lcSession.GetProperty(propertyToken)
```

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>propertyToken</i> | Long. Token representing the activity property. See Appendix B for a list of Tokens. |

Return Value

| <i>Value</i> | <i>Description</i> |
|------------------|--|
| <i>destField</i> | Value of the activity property. If the property value is of a different type than this field, then data conversion will occur. The property value will be written into the first value in the field. |

GetProperty<Type> Method for LCSession

The GetProperty<Type> methods return a Session property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

This method allows retrieval of Session properties as with LCSession.GetProperty, but does not require a field object. If the property value is of a different type, data conversion will be performed.

Defined In

LCSession Class

Syntax

flag = *lcSession*.**GetPropertyBoolean**(*propertyToken*, *default*)

Set destCurrency = *lcSession*.**GetPropertyCurrency**(*propertyToken*)

Set destDatetime = *lcSession*.**GetPropertyDatetime**(*propertyToken*)

destFloat = *lcSession*.**GetPropertyFloat**(*propertyToken*)

destInt = *lcSession*.**GetPropertyInt**(*propertyToken*)

Set destNumeric = *lcSession*.**GetPropertyNumeric**(*propertyToken*)

Set destStream = *lcSession*.**GetPropertyStream**(*propertyToken*, *streamFormat*)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>propertyToken</i> | A token identifying the Session property. See Appendix B for a list of Tokens. |
| <i>default</i> | (GetPropertyBoolean only) Value to be returned if the property cannot be located. Default value is FALSE. |
| <i>streamFormat</i> | (GetPropertyStream only) Format that the stream is converted to before being returned. If streamFormat is zero, no conversion occurs and the stream is copied in the same format as the property value. |

Return Values

| <i>Value</i> | <i>Description</i> |
|-------------------------|---|
| <i>flag</i> | (GetPropertyBoolean only) Boolean value, either TRUE or FALSE. If the property does not exist, the default is returned. |
| <i>dest<Type></i> | Current value for the Session property. |

ListProperty Method for LCSession

This method obtains the first or next property information for properties supported by the Session.

Defined In

LCSession Class

Syntax

Call *lcSession.ListProperty(list, propertyToken, dataType, propertyFlags, propertyName)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>list</i> | Long. Constant indicating whether to return the first or next Session property. LCLIST_FIRST Return the first property in the property list. LCLIST_NEXT Return the next property (or the first property if this is the first call to this method for this Session). |
| <i>propertyToken</i> | Long. Optional. Token assigned to the property. See Appendix B for a list of Tokens |
| <i>dataType</i> | Long. Optional. Data type of the property. |

continued

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>propertyFlags</i> | Property flags for the property. None defined at this time. Value is always zero. Optional. |
| <i>PropertyName</i> | String. Optional. Name of the property. |

LookupProperty Method for LCSession

This method determines if the specified property has been defined in the activity.

Defined In

LCSession Class

Syntax

flag = *lcSession.LookupProperty* (*propertyToken*, *dataType*, *propertyFlags*, *propertyName*)

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|---|
| <i>propertyToken</i> | Long. A token identifying the Session property. See Appendix B for a list of Tokens. |
| <i>dataType</i> | Long. Optional. The property data type. |
| <i>propertyFlags</i> | Long. Optional. The property flags from the following list. LCPROPERTYF_CONNECT Property is used for connecting. LCPROPERTYF_BOOLEAN Property is a boolean value. LCPROPERTYF_READONLY Property is read-only. LCPROPERTYF_TEXTLIST Property is a text list. |
| <i>propertyName</i> | String. Optional. The name of the property. |

Return Value

| <i>Value</i> | <i>Description</i> |
|--------------|---|
| <i>flag</i> | TRUE or FALSE, indicating whether the property is supported for this Session. |

Log Method for LCSession

This method adds a new entry to the log for this activity.

Use LCSession.Log to log LEI errors for which the error codes are defined by LEI. This Log function logs a basic error or event; use LogEx for extended logs and LogText or LogTextEx for external logs. When logging an error, the activity status is set, placing the activity into an error state. The log text corresponding to the status code is added to any Log entry for this activity.

Defined In

LCSession Class

Syntax

lcSession.Log (status)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| <i>status</i> | Long. LEI status code to log. Most status codes are errors; those ORed with the flag LCSTATUSF_EVENT are considered non-error events. |

LogEx Method for LCSession

This method adds a new extended entry to the log for this activity.

Defined In

LCSession Class

Syntax

lcSession.LogEx (status, int1,int2, int3, text1, text2, text3)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------|---|
| <i>status</i> | Long. Base LEI status code and also the return value from the function. |
| <i>int1</i> | Long. Optional Extended parameters. See Usage below. |
| <i>int2</i> | Long. Optional Extended parameters. See Usage below. |
| <i>int3</i> | Long. Optional Extended parameters. See Usage below. |
| <i>text1</i> | String. Optional Extended parameters. See Usage below. |
| <i>text2</i> | String. Optional Extended parameters. See Usage below. |
| <i>text3</i> | String. Optional Extended parameters. See Usage below. |

Usage

This Log method logs an extended error or event with parameters; use Log for basic logs and LogText or LogTextEx for external logs. When logging an error, the activity status is set, placing the activity into an error state. The log text corresponding to the status code is added to any Log entry for this activity.

Each of the supported extended errors is described below along with the extended *intN* and *textN* parameters used by that error:

| <i>Value</i> | <i>Description</i> |
|-----------------------------|--|
| LCFAIL_INVALID_METADATA | Metadata (<i>text1</i>) |
| LCFAIL_TYPE_MISMATCH | FieldName (<i>text1</i>), LCType (<i>Int1</i>), ConnectorType (<i>Int2</i>) |
| LCFAIL_DUPLICATE | Metadata (<i>text1</i>) |
| LCFAIL_FIELD_COUNT_MISMATCH | LCCount (<i>int1</i>), LinkCount (<i>Int2</i>) |
| LCFAIL_KEY_COUNT_MISMATCH | LCCount (<i>Int1</i>), LinkCount (<i>Int2</i>) |
| LCFAIL_STAMPFIELD_TYPE | FieldName (<i>text1</i>), LinkType (<i>Int1</i>) |
| LCFAIL_FIELD_TYPE | FieldName (<i>text1</i>), ExpectedType (<i>Int1</i>), ActualType (<i>Int2</i>) |
| LCFAIL_MERGE_FIELD | FieldName (<i>text1</i>) |
| LCFAIL_MISSING_PROPERTY | PropertyToken (<i>Int1</i>) |
| LCFAIL_PROPERTY_CONFLICT | PropertyToken (<i>Int1</i>), PropertyToken (<i>Int2</i>) |
| LCFAIL_INVALID_PROPERTY | PropertyToken (<i>Int1</i>) |
| LCFAIL_PROPERTY_VALUE | PropertyToken (<i>Int1</i>) |
| LCEVENT_CHARACTER_SET | Character Set Descriptor (<i>text1</i>) |

The following status values are in the error package LCXPKG_EXFIELD. All of these take a single extended parameter, which is the field name, in *Text1*:

LCFAIL_OVERFLOW

LCFAIL_PRECISION_LOSS

LCFAIL_INVALID_INT

LCFAIL_INVALID_FLOAT

LCFAIL_INVALID_CURRENCY

LCFAIL_INVALID_NUMERIC

LCFAIL_INVALID_DATETIME

LCFAIL_INVALID_STREAM

LCFAIL_INVALID_FIELD
LCFAIL_INVALID_TYPE
LCFAIL_INVALID_KEY
LCFAIL_DUPLICATE_KEY
LCFAIL_INVALID_STAMPFIELD
LCFAIL_INVALID_FIELDNAME
LCFAIL_VIRTUAL_FIELD
LCFAIL_VIRTUAL_VALUE
LCFAIL_INVALID_ORDER

LogText Method for LCSession

This method adds a new external entry to the log for this activity.

This Log method logs an external basic error or event; use LogTextEx for external extended logs and Log or LogEx for LEI errors. When logging an error, the activity status is set, placing the activity into an error state. A message is composed from the external text and code, and is added to any Log entry for this activity. Use the method to log external errors from data providers when LEI does not have the error message available. Obtain the error code and text for an external error message, and call LogText to add that information to the LEI log. The LEI error code used for this type of error is LCFAIL_EXTERNAL, which is the same value returned by the method for error logs.

Defined In

LCSession Class

Syntax

lcSession.LogText(*logFlags*, *text*, *externalCode*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>logFlags</i> | Long. Flags which affect this log entry. Zero or more of the following values are ORed together: LCLOGSTREAMF_EVENT Log a non-error event. |
| <i>text</i> | String. External log text. |
| <i>externalCode</i> | Long. External logcode. Optional. |

Example

'Create an LEI Scripted Activity that runs the script below.
LCStream text is written to the Scripted Activity Log.

Option Public

Useslx "lsxlei"

Sub Initialize

 'create the New LCSession and New LCStream Objects

 Dim myTestLog As New LCSession ("LogTest")

 Dim myTextString As New LCStream

 Dim x As Integer

 Dim y As Integer

 Dim z As Integer

 myTextString.text = "LEI 3.1 Script Example Using
 LCStream"

 x=1 ' For loop start value

 y=1 ' Counter start value

 z=10 ' For loop end value

 'write entries to the Scripted Activity log

 For x% = 1 To z

 Call myTestLog.Logtext (LCLOGSTREAMF_EVENT,
 myTextString.text & " -- Log Entry_"&Cstr(y), 0)

 y=y+1

 Next

End Sub

LogTextEx Method for LCSession

This method adds a new external extended entry to the log for this activity.

This Log method logs an external extended error or event; use LogText for external basic logs and Log or LogEx for LEI errors. When logging an error, the activity status is set, placing the activity into an error state. A message is composed from the external text, code, and additional parameters, and is added to any Log entry for this activity.

Defined In

LCSession Class

Syntax

lcSession.**LogTextEx**(*logFlags*, *text*, *externalCode*, *int1*, *int2*, *int3*, *text1*, *text2*, *text3*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>logFlags</i> | Long. Flags which affect his long entry. LCLOGSTREAMF_EVENT Log a non-error event. |
| <i>text</i> | String. External log text. See Usage below. |
| <i>externalCode</i> | Long. External log code. |
| <i>int1</i> | Long. Optional extended parameters. See Usage below. |
| <i>int2</i> | Long. Optional extended parameters. See Usage below. |
| <i>int3</i> | Long. Optional extended parameters. See Usage below. |
| <i>text1</i> | String. Optional extended parameters. See Usage below. |
| <i>text2</i> | String. Optional extended parameters. See Usage below. |
| <i>text3</i> | String. Optional extended parameters. See Usage below. |

Usage

The text in *text* for extended logs contains parameter substitution markers which indicate placement and formatting of extended parameters. These substitution markers are of the format “[*Lc**xx*]”, where “*xx*” can be any of the following values:

- In Integer value, from parameter Int<*n*>
- Tn Type constant LCTYPE_XXX, from parameter Int<*n*>
- Pn Property token, from parameter Int<*n*>
- On Object constant LCOBJECT_XXX, from parameter Int<*n*>
- Sn Stream text, from parameter text<*n*>
- X Stop indicator if all extended parameters are zero/null/empty
- X(Left bound to remove if next parameter is zero/null/empty
- X) Right bound to remove if next parameter is zero/null/empty

The method of variable substitution allows for relatively complex multiple-parameter log text, as well as grammar variations between languages. The following examples are the actual LEI resources for three extended status values to demonstrate how this formatting is used:

| <i>Value</i> | <i>Description</i> |
|-------------------------|--|
| LCFAIL_DUPLICATE | “Duplicate object[[LCX]] '[[LCS1]]’”. |
| LCFAIL_INVALID_METADATA | “Metadata object [[LCX()]]'[[LCS1]]' [[LCX]]does not exist”. |
| LCFAIL_TYPE_MISMATCH | “Type mismatch[[LCX()]] for field '[[LCS1]]'[[LCX]] [[LCX()]; LEI: [[LCT1]] [[LCX]] [[LCX()]], Database: [[LCT2]] [[LCX]]” |

RunActivity Method for LCSession

This method causes one or more LEI activities to be executed.

Defined In

LCSession Class

Syntax

Call *thisSession.RunActivity* (*name*, *runFlags*, *stopDatetime*)

Parameters

| <i>Value</i> | <i>Description</i> |
|---------------------|---|
| <i>name</i> | String. The name of the activity to run. Multiple activities may be specified by separating each with a semicolon (;) within the string. |
| <i>runFlags</i> | Long. Optional. Flags that affect how the activities run. Zero or more of the following values ORed together: LCACTRUNF_SYNCHRONOUS Runs the activity and any dependent activities synchronously. In this case, the function will return only when all activities in the list and their dependents have completed, or when <i>stopDatetime</i> has been reached. The flag value is 0X01. |
| <i>stopDateTime</i> | LCDatetime. Optional. If you run activities synchronously then <i>StopDateTime</i> indicates the time at which this method will return if the activities are not yet complete. |

SetProperty Method for LCSession

This method assigns the current value for an activity property.

Defined In

LCSession Class

Syntax

Call *thisSession.SetProperty (propertyToken, srcField)*

Parameters

| <i>Value</i> | <i>Description</i> |
|----------------------|--|
| <i>propertyToken</i> | Long. Token representing the Session property for which the value is to be assigned. See Appendix B for a list of Tokens. |
| <i>srcField</i> | LCField. New value for the Session property. If the property value is of a different type than this field, then data conversion will occur. The property value will be assigned from the first value in the field. |

SetProperty<Type> Method for LCSession

The SetProperty<Type> methods assign a property value as a specific data type. Datatypes supported include: Boolean, Currency, Datetime, Float, Int, Numeric, and Text.

Defined In

LCSession Class

Syntax

Call *lcSession.SetPropertyBoolean(propertyToken, srcBoolean)*

Call *lcSession.SetPropertyCurrency(propertyToken, srcCurrency)*

Call *lcSession.SetPropertyDatetime(propertyToken, srcDatetime)*

Call *lcSession.SetPropertyFloat(propertyToken, srcFloat)*

Call *lcSession.SetPropertyInt(propertyToken, srcInt)*

Call *lcSession.SetPropertyNumeric(propertyToken, srcNumeric)*

Call *lcSession.SetPropertyStream(propertyToken, srcStream)*

Parameters

| <i>Value</i> | <i>Description</i> |
|------------------------|--|
| <i>propertyToken</i> | Long. Token identifying the Session property. See Appendix B for a list of Tokens. |
| <i>src<Type></i> | LCStream. Value to assign to the Session property. |

Lotus

An IBM Company



Part No. CT7KQNA